Data efficient deep learning for automated visual environment monitoring

Geoff French

A thesis presented for the degree of
Doctor of Philosophy

School of Computer Science
University of East Anglia
United Kingdom
November 2023

Data efficient deep learning for automated visual environment monitoring

Geoff French

2023

Abstract

Deep neural networks have been used to establish a number of state of the art results in computer vision over the last several years. Their use has resulted in a step-change in performance in a variety of computer vision tasks, often raising accuracy over the threshold at which automated analysis becomes practically useful. This performance however comes at a cost requiring large labelled datasets for training.

Computer vision training sets typically consist of input images with corresponding ground truth annotations. Acquiring or producing them can be challenging. In some domains – e.g. medical imagery – the major difficulties are associated with acquiring input images due to the necessity of expensive equipment or complications due to privacy concerns. In many domains – particularly those involving photographic imagery – input images are readily available, while the manual process involved in producing the ground truth annotations acts as a bottleneck as it can be a laborious and expensive task.

In this thesis we will discuss practical computer vision problems where the cost of producing ground truth annotations poses a significant limitation. We explore and discuss a number of approaches aimed at reducing the quantity of ground truth labels required or reducing the effort required to produce them.

This work is motivated by practical problems related to environmental monitoring that will be discussed, along with the solutions that we were able to apply.

Data efficient deep learning for automated visual environment monitoring

Geoff French

Acknowledgements

I would like to thank my supervisory team, Prof. Graham Finlayson and Dr. Michal Mackiewicz for their support, advice, and help throughout my PhD.

We would like to thank Charlotte Atlass, Luisa Barros, Daniel Clarke, Lauren Clayton, Scott Connell, James Dooley, Liseve Fierens, Helen Holah, Kelly James, Rachel Kilburn, James Lamb, Emma Mackenzie, Anastasia Moutaftsi, Joseph Ribeiro, Louisa Sinclair, Rebecca Skirrow, Maria Wild and Hannah Wolstenholme at Marine Scotland Science and Cefas for annotating our images in order to produce training data for our fish quantification system discussed in Chapter 7.

We would like to thank Helen Holah at Marine Scotland Science for the idea of using single species footage captured on board their research vessel and for the idea of matching fish by estimated position rather than discard time (see Chapter 7).

Part of the work discussed in Chapter 5 was done during an internship at nVidia. Much of the computation required was performed on the University of East Anglia HPC Cluster; we would like to thank Jimmy Cross, Amjad Sayed and Leo Earl for their help in this regard. Some was also performed on the nVidia Helsink research cluster; we would like to thank Janne Hellsten and Samuli Laine.

We would like thank nVidia corporation for their generous donation of a Titan X GPU.

The work discussed in Chapter 6 was done during an internship at the Brain team, Google Research, Amsterdam. The computation required was performed using the Google Research cluster.

The work discussed in Chapter 3 was funded by Spectral Edge Ltd. The work discussed in Chapter 7 was funded under the European Union Horizon 2020 SMARTFISH project, grant agreement no. 773521.

Contents

List of Figures

List of Tables

1 Introduction

The widespread availability and low cost of cameras and data storage has resulted in the capture of vast quantities of imagery and video footage. The analysis of such imagery – whether the task involves identifying individuals in surveillance footage, or quantifying and identifying cells in microscope images – has become a bottleneck, as it was a manual and labour intensive task. Within the last several years, advances in computer vision have yielded approaches with sufficient accuracy to automate this laborious work.

Many of these advances utilise deep neural networks, a machine learning model that has established a number of state of the art results in computer vision over the last several years. Their use has resulted in a step-change in performance in a variety of tasks, frequently raising accuracy over the threshold at which automated analysis becomes practically useful. This performance however comes at a cost requiring large labelled datasets for training.

Computer vision training sets typically consist of input images with corresponding ground truth annotations. Acquiring or producing such data sets can be challenging. In some domains – such as medical imagery – the major difficulties are associated with acquiring input images due to the necessity of expensive equipment or complications due to privacy concerns. In many domains – particularly those involving photographic imagery – input images are readily available, while the manual process involved in producing the ground truth annotations acts as the bottleneck.

Creating ground truth annotations for a machine learning model is often more labour intensive than the original manual quantification task (on a per-image basis), often due to the additional detail required. For example, manually counting cells of different types present in microscopy images requires the observer to produce a per-cell type aggregate count. On the other hand, training a machine learning model to perform the equivalent task requires precise annotations that provide the location and type of each cell, each of which must be manually drawn by the observer. Analysing medical imagery may require an assessment of the presence or absence of certain structures or anomalies, whereas the ground truth training annotations are likely to require precise segmentation maps identifying the structures of interest. This highlights a dilemma that faces the development of automated visual quantification systems; the combined cost of producing the training data and developing the system may outweigh the costs saved by its use. This limits the application of automated visual monitoring to problems where large enough quantities of imagery must be analysed in order for the gain in efficiency to outweigh the upfront training cost.

The work discussed in this thesis was conducted with an environmental monitoring application in mind; the automated quantification of by-catch (discarded fish) from surveillance footage captured on-board fishing trawlers. Constraints arising from the aforementioned issues manifested early during the development of our CCTV footage analysis system. This motivated us to focus the majority of the research discussed in this thesis on

Figure 1.1: VGA images from the CCTV for Fisheries project

semi-supervised learning and unsupervised domain adaptation; two promising avenues of research that promise to alleviate the labelling bottleneck. As will be seen in Chapter 7 we successfully applied semi-supervised learning to improve the performance of our by-catch quantification system.

1.1 Automated by-catch quantification

The work discussed in this thesis informed the development of CatchMonitor (discussed in further detail in Chapter 7); a computer vision system designed for automated by-catch quantification from fishing trawler surveillance footage. Manual quantification requires an observer to identify the individual fish that are discarded (the fish retained for sale at port do not need to be considered here and can be ignored), identify their species and measure their on-screen length for the purpose of mass estimation. The manual process is supported by software designed to aid this work-flow.

The 'CCTV for Fisheries' project was a pilot study funded by Marine Scotland for the purpose assessing the feasibility of automated quantification of by-catch. The progress made during the project demonstrated that this is a plausible goal and established deep neural networks as the most promising approach for processing the challenging real-world footage, shown in Figure 1.1. CatchMonitor continued this work under the umbrella of SMARTFISH; a Horizon 2020 project funded by the European Union. Within Chapter 7 we make extensive use of data that was provided by partner members of the SMARTFISH project.

1.2 Proposed solution

During the development of CatchMonitor we identified the components that it would require:

1. An instance segmentation system to isolate each individual fish in a frame image extracted from a video
2. A species classifier to identify the species of each individual

3. An object tracking system to track the fish for the duration in which they are visible, in order to associate the frame-by-frame detections with one another. Without this each fish would be counted multiple times; once for each frame in which it appears.

The first phase of the 'CCTV for Fisheries' project focused on the instance segmentation component. Manual annotation tools were developed for the purpose of producing annotations to train the segmentation model. It quickly became apparent that annotation was a laborious and time consuming task, as each fish required a precise segmentation outline. The development of this component was continued and refined in the SMARTFISH project. Later phases focused on species identification. This task also required annotation for the purpose of producing ground truths. While some of the segmentation annotations could be produced by the author, species identification requires experienced marine biologists. The species annotation task also proved to be time consuming and labour intensive.

1.2.1 Reducing the annotation bottleneck

The annotation bottleneck was identified as a limiting factor in the early stages of the project. Different vessels used different materials for their conveyor belts and had different lighting conditions. The resulting variation in visual appearance hampered the performance of a model when given footage from a vessel that it was not trained on. As a consequence it was necessary to provide ground truth annotations for each vessel. This motivated us to explore the use of semi-supervised learning and efficient tools that assist the user during the annotation process.

Semi-supervised learning

Semi-supervised learning is an established area of research in machine learning that attempts to learn from a combination of labelled and unlabelled samples. It offers the tantalising possibility of improving the performance of a computer vision model by using unlabelled (un-annotated) samples – in addition to the existing annotated samples – during training. Successful semi-supervised classification algorithms [Laine and Aila, 2017; Tarvainen and Valpola, 2017] have achieved impressive performance while requiring ground truth labels for only a small fraction of the training samples.

A large quantity of training images were extracted from the CCTV footage that was captured during the course of the 'CCTV for Fisheries' and SMARTFISH projects. Producing ground truth annotating for the entire dataset proved to be infeasible. Semi-supervised learning therefore appeared to be a natural fit for fish by-catch quantification, as it would allow us to improve performance of our models by utilising the large quantities of un-annotated images, in addition to the small number that we were able to acquire annotations for.

Unsupervised domain adaptation

A machine learning model trained on supervised data from a source domain (e.g. artificial 3D rendered images) will often over fit and perform poorly on data from a distinct tar-

get domain (e.g. real photographic images). Unsupervised domain adaptation algorithms attempt to reduce the effect of this *domain shift*.

The differing appearance of footage obtained from different cameras in different locations frequently hampers the performance of computer vision model. This complicates the process of applying a computer vision model to footage obtained from a novel location for which there is no annotated training data. This problem arose during CatchMonitor as a result of the differing appearance of footage obtained from different vessels. An effective domain adaptation algorithm could be used to alleviate this problem.

Efficient annotation tools

The main cost of producing ground truth annotations is due to the human labour required. Efficient annotation tools maximise the effectiveness of the annotators by reducing the amount of time that must be spent on each sample. Improvements to annotation tools can take the form of effective user interfaces and automated assistance to hasten the annotation process.

1.3 Contributions

In this thesis we will cover five contributions. Our first four key contributions provide context for the design of components in the fifth; the CatchMonitor system. We will summarise them here, along with our publications that arose during their development.

Multi-spectral object detection. We explore the use of spectral edge fusion [Connah et al., 2015] as a pre-process to fuse RGB and thermal images prior to processing using an object detection network, with a focus on pedestrian detection. We compare the efficacy of image fusion with using a network architecture that directly accepts multi-spectral input. Our use of image fusion highlights issues caused by misalignment between RGB and thermal channels that arise when the images are captured using separate cameras. This work was published as "Multi-spectral Pedestrian Detection via Image Fusion and Deep Neural Networks", by French, Finlayson and Mackiewicz at the Color Image Conference, 2018 [French et al., 2018a].

Consistency regularization for unsupervised domain adaptation. Approaches based on consistency regularization [Oliver et al., 2018] established some of the earlier [Laine and Aila, 2017] state of the art semi-supervised classification results. We contribute an adaptation of consistency regularization to work in unsupervised domain adaptation. We established new state of the art (as of April 2018) results in unsupervised domain adaptation for standard benchmarks and achieved 1st place in the VisDa 2017 domain adaptation competition [Peng et al., 2018]. This work was published as "Self-ensembling for visual domain adaptation", by French, Mackiewicz and Fisher at the International Conference on Learning Representations, 2018 [French et al., 2018b].

CutMix for semi-supervised semantic segmentation. Continuing our exploration of semi-supervised learning we adapted CutMix regularization [Yun et al., 2019] for semantic segmentation. We contribute an analysis of the semantic segmentation problem in which

we discovered that the input data distribution exhibits properties that make it particularly challenging from the stand point of consistency regularization for semi-supervised learning. Our adapted CutMix regularizer overcomes this, achieving state of the art results. This work was published as "Semi-supervised semantic segmentation needs strong, varied perturbations" by French, Laine, Aila, Mackiewicz and Finlayson at the British Machine Vision Conference, 2020 [French et al., 2020a]. Our work on the use of colour augmentation to obtain improved results was published as "Colour augmentation for improved semi-supervised semantic segmentation" by French and Mackiewicz at the VISAPP Conference, 2022 [French. and Mackiewicz., 2022].

Milking CowMask for semi-supervised image classification. Following our use of mask-based regularization for semi-supervised semantic segmentation, we use this approach for image classification. We contribute a mask-based regularizer called CowMask, so called due to its Friesian cow-like appearance. We achieved results in 10% semi-supervised ImageNet in January 2020 that were state of the art at the time. This work is to be published as "Milking CowMask for Semi-Supervised Image Classification" by French, Oliver, and Salimans at the VISAPP Conference, 2022 [French. et al., 2022].

Deep neural networks for by-catch quantification. We contribute the design of a system for the automated quantification of by-batch from fisheries surveillance video. It utilises instance segmentation to isolate each individual fish in an image, image classification to determine the species of each individual and object tracking to track individuals while they are visible. The system generates a per-species count of individuals that are discarded. We also developed a web-based annotation tool to allow marine biologists to outline and identify the species of fish in images in order to provide training data. An earlier version of this work was published as "Deep neural networks for analysis of fisheries surveillance video and automated monitoring of fish discards", by French, Mackiewicz, Fisher, *et al.* in the ICES Journal of Marine Science, 2020 [French et al., 2020b].

1.4 Thesis structure

This thesis is structured as follows; we start by presenting a survey of the background literature, with the following central chapters focusing on the experiments and novel techniques whose results support the components used to build the overall system described in the penultimate chapter, after which we present our conclusions and discuss future work.

We present the background work related to the areas covered in this thesis in Chapter 2. We survey a number of areas in computer vision, with a strong focus on the deep learning based classification, instance segmentation and semi-supervised learning areas that we utilise in this project. We also cover classic computer vision techniques that we used for estimating the motion of the conveyor belt and for object tracking in Chapter 7.

In Chapter 3 we discuss our comparison of spectral edge fusion [Connah et al., 2015] with multi-spectral network architectures. Multi-spectral cameras able to capture infra-red or thermal imagery are frequently used in environmental monitoring applications where low light levels or other adverse conditions would limit the effectiveness of a visible light camera. Furthermore this work was an important step along the journey as it helped us develop our

understanding of neural network based object detection systems and the issues surrounding the use of multi-spectral imagery.

Chapter 4 describes our adaptation of a consistency regularization based semi-supervised learning algorithm for unsupervised domain adaptation. As stated earlier, domain adaptation has the potential to be very beneficial for environmental monitoring problems such as CatchMonitor, given the domain gap that exists between the visual appearances of footage obtained from different locations.

In Chapter 5 we explore the problem of semi-supervised semantic segmentation using consistency regularization. Semantic segmentation has a variety of applications in environmental monitoring, often focused on quantifying regions of interest within an image, *e.g.* forest coverage in satellite imagery or disease quantification within images of vegetation. The work discussed in this chapter also yielded very useful insights into the area of semi-supervised learning.

We propose CowMask in Chapter 6, along with workable approaches for semi-supervised image classification. CowMask was evaluated for the purpose of species identification in CatchMonitor. The work discussed in this chapter was very helpful in developing our knowledge of semi-supervised learning and informed the use of semi-supervised learning in CatchMonitor.

Chapter 7 presents our by-catch quantification system. We discuss our design decisions and their motivation and evaluate the effectiveness of our system.

Finally, in Chapter 8 we present our conclusions and discuss avenues for future work.

1.5 Publications

For convenience we list the publications that form the backbone of this thesis here:

- French, G., Finlayson, G., and Mackiewicz, M. Multi-spectral pedestrian detection via image fusion and deep neural networks. *Color and Imaging Conference*, 2018.
- French, G., Mackiewicz, M., and Fisher, M. Self-ensembling for visual domain adaptation. *In International Conference on Learning Representations (ICLR)*, 2018.
- French, G., Laine, S., Aila, T., Mackiewicz, M., and Finlayson, G. Semi-supervised semantic segmentation needs strong, varied perturbations. *In Proceedings of the British Machine Vision Conference (BMVC)*, 2020.
- French., G. and Mackiewicz., M. Colour augmentation for improved semi-supervised semantic segmentation. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, 2022.
- French., G., Oliver., A., and Salimans., T. Milking CowMask for semi-supervised image classification. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, 2022.

- French, G., Mackiewicz, M., Fisher, M., Holah, H., Kilburn, R., Campbell, N., and Needle, C. Deep neural networks for analysis of fisheries surveillance video and automated monitoring of fish discards. *ICES Journal of Marine Science*, 2020.

2 Background

In this chapter we discuss relevant literature that we used to inform and direct our research. We start by introducing deep learning and computer vision with a discussion of the image classification models that propelled deep neural networks to the forefront of computer vision research.

The convolutional stems of state of the art classification models were re-purposed for a variety of computer vision problems including object detection and segmentation. This process – known as *transfer learning* – improves accuracy and significantly simplifies the training of complex models. We discuss transfer learning, followed by deep-learning based object detection and segmentation models

The practical scenarios that are the motivation for the work in this thesis drew our attention to the annotation bottleneck. This motivated us to explore approaches for using out limited training data more effectively.

Data augmentation as it is an important technique for reducing over-fitting of machine learning models. We discuss a variety of approaches that have been developed for computer vision problems .

Approaches that learn from un-labelled training data could be highly beneficial for the problems that we focus on. We explore semi-supervised learning, domain adaptation and self-supervised learning techniques with a view to obtaining the maximum accuracy from that limited amount of labelled training data that we have. We also briefly discuss active learning as it has the potential to be practically useful.

Finally we discuss the techniques that enabled components of CatchMonitor – some of which are based on classic computer vision – including feature extraction and matching, object tracking and fish classification.

2.1 Image classification

Deep neural networks lie behind the state of the art image classification results that have been set over the last decade. The ImageNet [Russakovsky et al., 2015] classification challenge is the benchmark that is most commonly used to evaluate image classifiers. Krizhevsky et al. [2012] achieved a state of the art result, reducing the top-5 validation error rate to 18.2% (with a single network), from around 25% that can be achieved with classic computer vision techniques. Their architecture is commonly referred to as *AlexNet*. Since the introduction of AlexNet a number of new architectures have been developed that further improve accuracy.

Simonyan and Zisserman [2014] introduced the VGG architecture. They replaced convolutional layers with large kernels used in prior work with blocks of 3×3 convolutional layers.

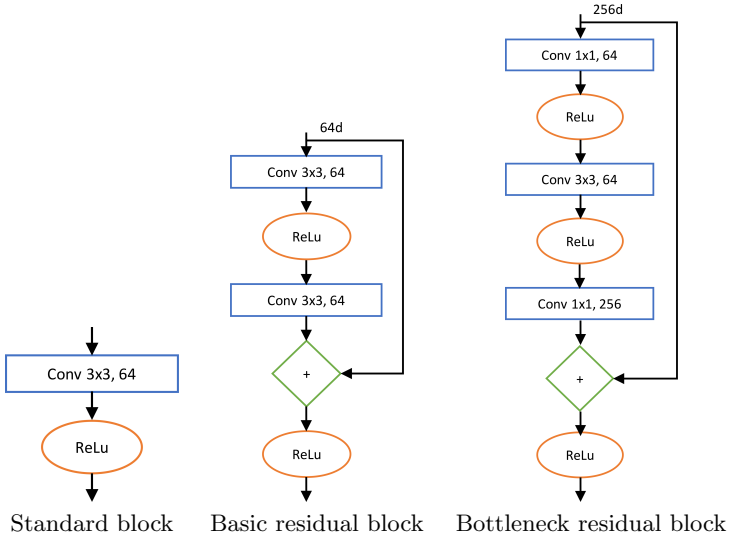


Figure 2.1: Standard and residual blocks

These blocks have the same receptive field while requiring fewer parameters and computational operations and offer increased accuracy. Their 19-layer VGG-19 architecture achieved a top 5 validation error rate of 7.5%. Blocks of 3×3 convolutional layers have become a commonly used architectural pattern since their introduction due to their effectiveness and simplicity.

Batch normalization [Ioffe and Szegedy, 2015] permits deeper networks to be trained in an end-to-end fashion, as training networks much deeper than around 8-10 layers is often unstable if batch normalization is not used. Simonyan *et al.* had to train their VGG networks in two stages; the earlier layers were pre-trained, before attaching the later layers.

He et al. [2016] introduced residual networks (ResNets) that utilise residual blocks. Their structure is contrasted with standard convolutional layers in Figure 2.1. The skip connection in a residual layer acts as the identify function, causing the convolutional layers to learn to produce a *residual* offset that is added to the identity function. ResNets consisting of hundreds of layers have been successfully trained, with He's 152 layer model bringing the top-5 ImageNet validation error rate down to 4.5%.

2.2 Transfer learning

Transfer learning describes the process in which a deep neural network is adapted for a different task. The most common approach involves adapting a network pre-trained for ImageNet [Russakovsky et al., 2015] image classification for a different purpose. The VGG-16 [Simonyan and Zisserman, 2014], ResNet-50 [He et al., 2016] or ResNet-101 networks are commonly used for this purpose. Early work [Donahue et al., 2014] demonstrated that the high level convolutional features extracted by the AlexNet [Krizhevsky et al., 2012] network could be used in place of features extracted using classic computer vision algorithms (e.g. HOGs [Dalal and Triggs, 2005]) for the purpose of training a classifier – such as a linear SVM

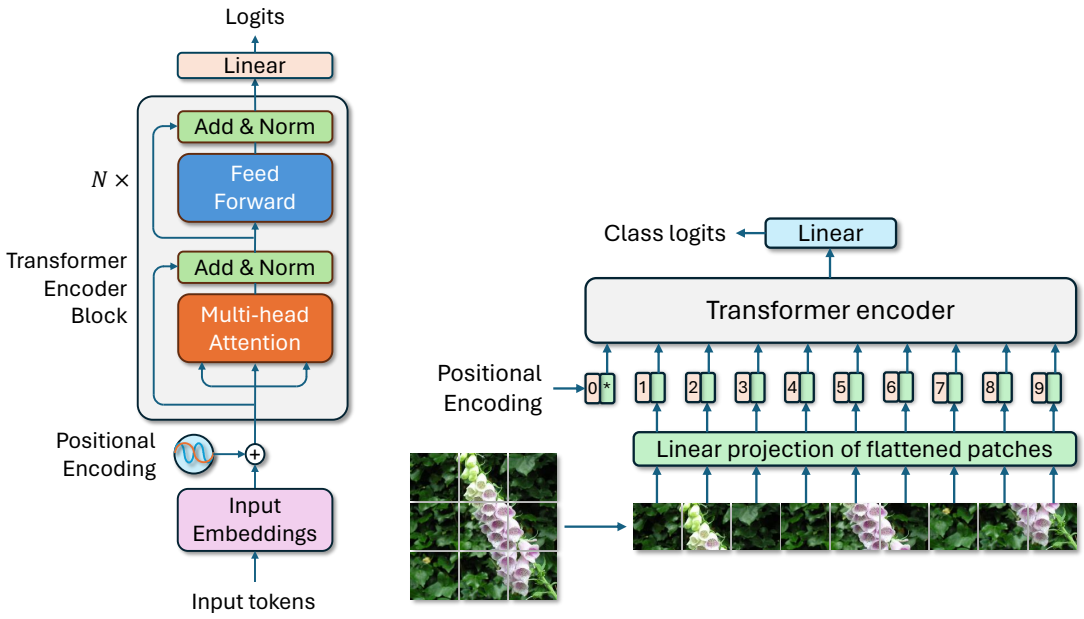
– for a different classification task. Long et al. [2015a] developed this approach further; they discarded the later fully connected layers of the VGG-16 network and replaced them with new layers that outputs a semantic segmentation in the form of pixel-wise classification. They found that the best results could be obtained by fine-tuning, in which the pre-trained layers are trained using a lower learning rate (usually $\frac{1}{10}\times$). Transfer learning has been successfully used in many computer vision tasks, including object detection and image segmentation.

He et al. [2018] reported that with sufficient training data, object detection results obtained without using transfer learning match those obtained when transfer learning is used, at the cost of longer training times. They conclude that while it considerably speeds up training, transfer learning is not necessary to achieve good results when sufficient training data is available.

2.3 Vision transformers

Dosovitskiy et al. [2021] introduced the vision transformer (ViT) architecture, an image classification model that adapted the transformer architecture [Vaswani et al., 2017] which was originally developed for natural language processing. The transformer architecture operates on an unordered set of input tokens combined with a per-token positional encoding that describes the sequential arrangement. When used for natural language processing the words – or part thereof – are used as tokens and used to select a corresponding input embedding from a lookup table. The positional encoding is designed to represent the sequential order of the tokens. The ViT architecture divides an input image into a grid of patches, each of which is linearly projected to produce a token, while a 2D positional encoding represents the grid position of each token. In contrast to convolutional architectures, the attention mechanism used in transformers allows the model to consider the relationship between all pairs of tokens across the entire image, as opposed to only local receptive fields. An illustration of the transformer encoder architecture that is the basis of the vision transformer is shown in Figure 2.2(a), with the vision transformer illustrated in Figure 2.2(b). Since 2020 transformer based models have become the dominant architecture across many modalities, including natural language processing, speech recognition and computer vision.

A basic vision transformer trained only on ImagenNet [Russakovsky et al., 2015] achieved accuracy slightly below that of a ResNet of comparable size. Vision transformers became more competitive [Dosovitskiy et al., 2021] when pre-trained on larger datasets, such as the JFT-300M dataset [Hinton et al., 2015; Chollet, 2017], consisting of 300 million images. More recent work has obtained competitive results when training a vision transformer from scratch on ImageNet [Beyer et al., 2022] without the need for pre-training; Beyer et al. [2022] used simple modifications to the training scheme and rich data augmentation while Chen et al. [2022] used the sharpness aware minimization (SAM) regularization technique [Foret et al., 2021]. The recent Swin transformer [Liu et al., 2021] architecture uses a hierarchical structure and achieved strong ImageNet classification results. Its self-attention mechanism is limited to local non-overlapping windows, while using shifted windows in alternating transformer blocks to allow communication between windows. Patch merging in effect downsamples the resolution of the patch grid, allowing the model to consider the relationship



(a) Transformer encoder (b) Vision transformer (ViT)

Figure 2.2: Transformer encoder and vision transformer

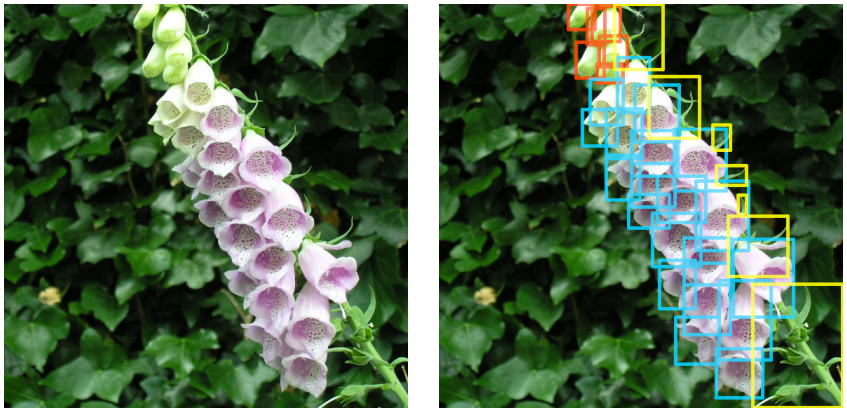
between tokens at increasing scales. The repeated local attention followed by down-sampling bears similarity to the convolution and pooling operations used in CNN based models.

We would like to note that the lions share of the developments that have occurred in vision transformers were published after the completion of the work covered in this thesis.

2.4 Object detection

Object detection is the task of localizing and classifying objects within an image, with the output consisting of a list of bounding boxes and corresponding class labels, one for each object. It is illustrated in Figure 2.3.

Deep neural network based object detection techniques were initially developed by gradually replacing components of classic computer vision based object detection algorithms. The classic approach of Dalal and Triggs [2005] used histogram of oriented gradients (HOGs) as a feature extractor to generate a feature description of a region of an image. These features would be passed to a classifier that would determine if an object was present in the image region. This region classifier would be applied in a sliding window fashion across the image, often at a variety of scales and aspect ratios. Sliding window based approaches incur significant cost due to the large number of regions that must be classified. This cost can be reduced using approaches such as selective search [Uijlings et al., 2013] that propose a smaller number of image regions that are likely to contain objects and classifying only these. Non-maximal suppression is often used as a final step to eliminate or reduce instances in which an object is detected multiple times. This is normally done by



(a) Input image (b) Object detection boxes

Figure 2.3: Object detection. The three colours of the boxes in (b) indicate the class of the object; red for un-opened flower, blue for flower and yellow for stem.

identifying pairs of detections in which there is significant area overlap – normally measured using area of intersection over union (IoU) – and eliminating the detection with the lowest confidence.

R-CNN (regions with CNN features) [Girshick et al., 2014] used selective search to propose regions and replaced the HOGs feature extractor with an AlexNet [Krizhevsky et al., 2012] based classifier. The image regions were scaled to 227×227; the input resolution of AlexNet. The last layer of AlexNet – a 1000-unit fully-connected classification layer – was removed and replaced with a 21-unit classification layer, corresponding to the 21 classes in the PASCAL VOC 2012 dataset [Everingham et al., 2012]. The network was then fine-tuned using images extracted from PASCAL VOC. Finally, linear SVM classifiers were trained to classify objects within image regions from the 4096-dimensional feature vector generated by the penultimate layer of the network. Localisation accuracy was further improved by training a regressor to predict bounding box refinements to improve the accuracy of detections further.

SPPnet [He et al., 2014] based object detectors run significantly faster than those based on R-CNN. R-CNN extracts regions from the image and passes each one through an AlexNet network, incurring significant computational cost as a result. In contrast, SPP-Net uses the convolutional layers to transform the complete original image into a lower resolution high-level feature image. Regions are then extracted from this high level feature image prior to being passed to a classifier. As a consequence, the computations incurred by the convolution operations used to generate high-level features are shared among all regions.

Fast R-CNN [Girshick, 2015] further improved on R-CNN and SPPnet by using a multi-task classification and regression (for bounding box refinement) during training and fine-tuning the network in an end-to-end fashion.

Faster R-CNN [Ren et al., 2015] replaces selective search – the last non-neural network based component – with region proposal networks (RPN); a fully convolutional network that proposes bounding boxes that are classified and refined by a Fast R-CNN module. The

result is a multi-head network with a shared backbone that consists of the convolutional layers of VGG-16. The RPN head is a fully convolutional network that generates predictions for each cell in the feature image generated by the VGG-16 backbone. Each cell corresponds to 32×32 pixels in the input image. To each cell are attached a number of anchor boxes of varying sizes and aspect ratios. For each anchor box the RPN predicts an *objectness* score – the likelihood that an object overlaps the anchor – and bounding box deltas that modify the size and position of the anchor to make it more closely match the bounding box of a detected object. Non-maximal suppression is used to reduce overlap and select the boxes that lie at the local maxima of predicted object probability (objectness). Faster R-CNN is a two-stage algorithm; the RPN generates box proposals that are passed to an RCNN head that uses bilinear filtering to crop a 7×7 feature image and classify the object and predict further box refinements. While Faster R-CNN is one of the most accurate object detection algorithms, the use of the R-CNN head used in the second stage incurs a significant performance penalty. More recent work often replaces the VGG-16 backbone with that of a more modern network such as ResNet-101.

You Only Look Once (YOLO) [Redmon et al., 2016; Redmon and Farhadi, 2017] and Single-shot detectors (SSD) [Liu et al., 2016; Fu et al., 2017] are fully-convolutional networks that detect objects in a single step. Both are structurally similar to region proposal networks (RPN) within Faster-RCNN, except that object detection is performed in one single step; the R-CNN based refinement step is omitted. As with RPN, the backbone (e.g. VGG-16 or ResNet-50) generates a feature image whose cells correspond to 32×32 pixels in the input image. For each cell, YOLO yields a class probability vector along with B (usually $B = 2$) box predictions, each consisting of a confidence score and box co-ordinates. During training, ground truth boxes are assigned to the cell that contains the box centre and matched with the anchor box that has the greatest intersection area over union area (IoU). The confidence score is trained to predict the IoU value. At inference time, the box for each cell with the highest confidence is selected and emitted. Non-maximal suppression can be used to further reduce overlap. YOLO can only predict one box for each cell, limiting its ability to distinguish objects that are close to one another in the image.

An SSD network is almost identical to that of an RPN, except that an SSD predicts a class probability vector instead of an *objectness* score. It is worth noting that the extreme class imbalance between background and foreground samples inherent in object detection scenarios hampers the performance of SSD and RPN networks. The original RPN network selected a balanced subset of samples from each image during training to counteract this. Lin et al. [2018] introduced focal loss that modulates standard cross-entropy loss with a term that gives additional weight to samples that are difficult to classify, achieving accuracies that can match those of two-stage detectors such as Faster R-CNN.

The object detection systems discussed so far are driven by features at a single resolution; $\frac{1}{32}$ of the resolution of the input image. This reduces the effectiveness of the model when detecting small objects. Similar problems affected classic computer vision (non-deep learning) based object detectors, leading to the use of image pyramids. An image pyramid is built by downscaling the image by a range of scale factors, e.g. 1, $\frac{1}{2}$, $\frac{1}{4}$, etc. The object detection model is trained and evaluated at all scales. This requires that each image in

the pyramid is processed separately, incurring significant computational cost when using a deep neural network as the feature extraction layers must be applied multiple times to the same image. We can take advantage of the fact that deep neural networks used for feature extraction (e.g. the VGG-16 or ResNet backbones) downscale internal feature representations several times within the network; attaching 'side taps' prior to each down-sampling operation allows the network to extract features at multiple resolutions in a single pass. Utilising only the highest resolution features as-is would result in sub-par performance as the image has only be passed through the earlier layers of the network, resulting in features that lack large scale context. Feature Pyramid Networks (FPNs) [Lin et al., 2017] progressively up-sample richer, later, lower resolution features and combine them with shallower, earlier, higher resolution features getting the best of both worlds. The structure of FPN shares similarities with the encoder-decoder architectures of the SegNet [Badrinarayanan et al., 2015] and U-Net [Ronneberger et al., 2015] segmentation networks. FPN based object detectors substantially improved accuracy in comparison to approaches that used features of a single resolution.

The work discussed in this section provides a variety of effective approaches for object detection that vary in speed and accuracy. They do however suffer from one flaw in that they cannot see *the elephant in the room*, literally. Rosenfeld et al. [2018] demonstrate this by inserting an image of an elephant into an image of a room and noting that modern object detectors fail to detect the elephant. The training data used for an object detector will consist of unedited photographs in which elephants will mostly appear against the background of their natural habitat or within a zoo. As a consequence the object detector will learn that an elephant is large grey animal against a background of consisting of grass and a landscape; removing these contextual cues degrades performance at inference time.

2.4.1 Pedestrian detection

Much of the early work on object detection was focused on pedestrian detection [Dalal and Triggs, 2005]. The ongoing development of self-driving cars underlies the continuing interest in this field of research.

Zhang et al. [2016] report that the R-CNN classification and bounding box refinement stage of a Faster R-CNN network hampered pedestrian detection performance in comparison to the underlying region proposal network (RPN) whose predictions it refines. They proposed a model that combines a RPN with a boosted forest based classifier instead of an additional neural network head. König et al. [2017] adopted this approach and proposed a multi-spectral RPN for pedestrian detection.

The KAIST multi-spectral pedestrian detection dataset was introduced by Hwang et al. [2015], along with a baseline detector based on aggregated channel features [Dollár et al., 2014]. The results obtained from their baseline approach clearly illustrated the effectiveness of multi-spectral imagery for pedestrian detection.

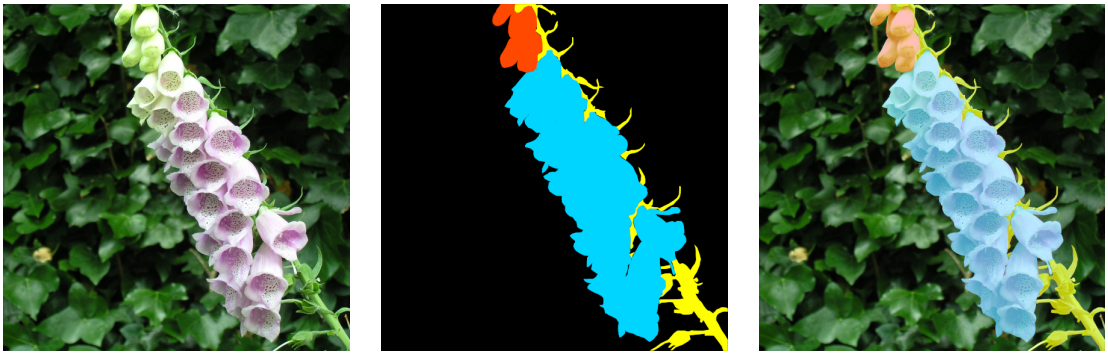
Jingjing Liu and Metaxas [2016] implemented a multi-spectral pedestrian detector based on a Faster R-CNN deep neural network object detector. They explored the effect of fusing RGB and thermal imagery at four different points within the network. Like the original

Faster R-CNN model, they used the VGG-16 [Simonyan and Zisserman, 2014] ImageNet classifier as a backbone. Their early and halfway fusion models fused the RGB and thermal paths within the convolutional stages of the network using Network-in-Network (NIN) [Lin et al., 2013] layers to blend the RGB and thermal features. The early and halfway models fused after the first and fourth block of convolutional layers respectively. Their late fusion model fused after the last feature generation layers, while their score fusion model averaged score predictions. Their halfway fusion model yielded the best performance.

2.4.2 Object detection with transformers

Since 2020 transformer based models have made significant in-roads into object detection. Carion et al. [2020] introduced the detection transformer (DETR) architecture that uses a transformer encoder-decoder model to predict the set of objects detected in an image, given image features as input generated by a CNN or transformer based vision backbone. As stated above, prior detection models – such as SSD [Liu et al., 2016], YOLO [Redmon et al., 2016] and Faster R-CNN [Ren et al., 2015] – make predictions with respect to a fixed set of anchor boxes, with each prediction indicating the presence of an object and giving alterations to the position and scale the bounding box of the corresponding anchor box. The thousands of predictions that typically result from a single image are filtered to retain only those that are predicted as belonging to a foreground class, after which non-maximal suppression eliminates duplicate detections. DETR instead directly predicts a set of detections. The transformer decoder receives a fixed-size set of N query embeddings that the self-attention mechanism compares and mixes with the input image features generated by the backbone. The decoder yields a set of N predictions, where N is set to be significantly larger than the typical number of objects in an image. Each prediction consists of a class probability vector – with an additional class representing *no detection* – and a bounding box. At training time the Hungarian algorithm is used to match ground truth objects to predictions, driven by a match cost function that considers both object class and bounding box similarity.

The deformable DETR model Zhu et al. [2021] addressed two limitations of the original DETR model; its slow convergence and its poor performance on small objects. For a given query the self-attention mechanism in DETR allows it to consider the relationship with all the input feature vectors generated by the backbone. During training, the self-attention mechanism within DETR must learn to focus on only the feature from the relevant parts of the image in order to detect an object. This is part of the reason for the slow convergence of the original DETR model. The deformable attention mechanism in deformable DETR draws inspiration from the deformable convolutions [Dai et al., 2017a]. For a given query it predicts a set of K offsets that are added to a reference point p_q, resulting in a set of K query points that are used to sample the input feature vectors using bilinear interpolation. The set of K sampled feature vectors are then used as keys for the self-attention mechanism; each query chooses the locations of the relevant parts of the image that it should consider. Due to the use of bilinear sampling, deformable DETR is designed to operate on image features generated by a convolutional backbone that preserved the grid structure of the input image.



(a) Input image (b) Semantic segmentation map (c) Image + segmentation map

Figure 2.4: Semantic segmentation. The three colours in the segmentation map in (b) indicate the class of the pixel; red for un-opened flowers, blue for flowers and yellow for stems. The segmentation map is overlaid on the input image in (c).

The aforementioned Swin transformer [Liu et al., 2021] and the improved Swin Transformer V2 [Liu et al., 2022] architectures yielded strong object detection performance due to the effectiveness of their hierarchical structure at focusing on regions of an image at different scales.

2.5 Segmentation

Segmentation is the process by which an image is separated into regions, usually on a per-pixel basis. Semantic segmentation systems can be seen to operate as pixel classifiers, predicting the class of the object or material type that covers each pixel in an image. The task of semantic segmentation is illustrated in Figure 2.4. Contour detection systems perform a complementary task; they identify pixels that lie on boundaries within an image, separating it into regions that correspond to different objects or materials. Instance segmentation system identify different objects within an image and predict the boundary that surrounds each one. Unlike semantic segmentation systems they are able to separate touching or overlapping instances of the same class.

2.5.1 Semantic segmentation

Many of the transfer learning based segmentation algorithms reported in the literature over the last few years are derived from fully convolutional networks [Long et al., 2015a]. A deep neural network pre-trained for ImageNet [Russakovsky et al., 2015] image classification – such as VGG-16 [Simonyan and Zisserman, 2014] or ResNet-101 He et al. [2016] – is adapted for segmentation by discarding its later layers (usually the fully-connected layers leading to the final layer that predicts image class probabilities) and replacing them with new randomly initialized layers that generate pixel-wise class predictions across the image.

Long et al. [2015a] introduced the fully convolutional network, demonstrating the effectiveness of deep neural networks for segmenting the PASCAL VOC 2012 dataset [Everingham et al., 2012]. They attached 21-way classifiers to the `pool3`, `pool4` and convolutionalized

`fc7` layers of the VGG-16 [Simonyan and Zisserman, 2014] ImageNet network, resulting in class predictions at 1/8th, 1/16th and 1/32nd resolution respectively. The `fc7` predictions are up-scaled by a factor of 2 using bi-linear filtering (performed using de-convolution operations) and combined with the predictions from `pool4`, which are in turn up-scaled and combined with those from `pool3` which are finally up-scaled by a factor of 8, resulting in a full-resolution pixel-wise class prediction for the image. They call this architecture *FCN-8s*. They train using mini-batches composed of a single complete image.

Further work built upon fully convolutional networks by modifying the architecture. Chen *et al.* simplified the FCN-8s architecture by using dilated/atrous convolutions [Chen et al., 2014] in the later layers of VGG-16, increasing the spatial resolution of their predictions while maintaining their receptive fields. The final 1000-way class prediction layer was replaced with a 21-way layer that produces predictions at 1/8th resolution that are up-sampled with bi-linear filtering to obtain full-resolution predictions.

Badrinarayanan *et al.* used VGG-16 as the encoder part of an encoder-decoder network [Badrinarayanan et al., 2015]. The decoder network is in effect a mirror image of the encoder, with each encoder layer having an equivalent decoder layer, in opposite order. Encoder-decoder architectures draw data from intermediate layers in the decoder to assist in the decoding process. This adds fine detail as later layers in the encoder tend to represent high level features rather than precise detail. The pooling indices from the VGG-16 max-pooling layers of the encoder drive equivalent un-pooling layers in the decoder.

U-nets [Ronneberger et al., 2015] have a similar architecture; except that strided deconvolution layers are used to up-sample complete feature maps drawn from intermediate layers within the decoder instead of un-pooling layers and complete feature maps rather than pooling indices are carried from intermediate encoder layers to the equivalent decoder layers. U-nets were developed within the medical image segmentation community were trained from scratch rather than using transfer learning. Badrinarayanan *et al.* noted that carrying complete feature maps results in slightly higher accuracy at the cost of longer runtime and higher memory usage. Their strong performance however has led to widespread adoption.

Zhao et al. [2017] improved segmentation accuracy by using a pyramid pooling module to extract large scale contextual features that are used in conjunction with ResNet features by the network's segmentation head. The contextual features provide the segmentation head with additional information on the surroundings of a particular region of the image, improving the accuracy of the segmentation of objects against common backdrops, e.g. a pillow lying on a bed or a boat against a backdrop of water.

The DEXTR model of Maninis et al. [2018] addresses the related problem of optimizing the work-flow used to generate the manual annotations required to train segmentation models. Manually drawing precise labels to provide outlines for objects within an image is a time consuming process. DEXTR partially automates this work-flow by predicting a mask covering an object, given four extreme points chosen the user. The extreme points are positioned on the top, bottom, left and right edges of the mask of the object. A region covering the bounding is extracted from the image and scaled to a fixed size. A

corresponding heat map of the same fixed size is generated, with a Gaussian blob centred on each extreme point. DEXTR modifies a standard segmentation model – DeepLab V2 [Chen et al., 2017] in the original implementation – to accept four input channels (R, G, B and heat map) and predict a class agnostic instance mask.

Conditional random fields

The use of conditional random fields (CRFs) for segmentation has a rich history in the literature [Russell et al., 2009; Krähenbühl and Koltun, 2011] prior to the wide adoption of deep learning. When used for semantic labelling, CRFs perform probabilistic inference and incorporate assumptions such as class agreement between similar or neighbouring pixels [Zheng et al., 2015]. CRF based models typically consist of a unary per-pixel probability prediction – often the prediction from a classifier – and a pairwise edge potential term that encourages similar predictions for similar pixels. Krähenbühl and Koltun [2011] classify TextonBoost [Shotton et al., 2009] based features extracted from an image, resulting in per-pixel class probabilities that are used as the unary term. These predictions alone produce a segmentation with a noisy appearance. The pairwise edge potential component of the CRF can be seen to smooth the classifier predictions within similarly coloured regions, while encouraging semantic segmentation boundaries to align to edges in the RGB image.

Chen et al. [2014] use CRFs to refine the segmentation output of their network. Their network outputs predictions that have a smooth and blurry appearance. Their CRF iteratively refines the segmentation, causing boundaries to align with boundaries in the RGB source image while encouraging smoothness in smoother areas of the underlying image. Zheng et al. [2015] formulate the CRF training and inference process as recurrent layers of a neural network, resulting in a model that can be trained end-to-end.

2.5.2 Contour detection

Contour detection algorithms are frequently evaluated using the BSDS-500 dataset [Martin et al., 2001]. It consists of 500 images; 200 for training, 100 for validation and 200 for testing. Each image has several ground truth annotations, prepared by different human annotators. The annotations come in the form of contour maps and the regions that result from splitting the images using the contour maps. Some algorithms are also evaluated using the Pascal VOC [Everingham et al., 2012] dataset.

The contour detection approaches discussed here can be divided into two categories; those that directly predict if a pixel lies on a contour and those that predict an 'edge map patch' at regularly spaced points (either at full resolution or a fraction thereof). The edge map patches usually significantly overlap and are therefore blended to produce a final edge map. The edge map patch approaches will be discussed first.

Dollár and Zitnick [2013] developed an effective classical computer vision based contour detection algorithm that used structured forests to select an edge map patch from a dictionary, given an input RGB image patch. The patches are blended together to produce the final predicted contour map. Ganin and Lempitsky [2014] replaced the structured forest

(a) Input image (b) Instance segmentation map (c) Image + segmentation map

Figure 2.5: Instance segmentation. The colours in (b) indicate the separate object instances. The segmentation map is overlaid on the input image in (c).

with a deep neural network regressor to predict a codeword that is used to choose a contour patch from a dictionary. The edge patch dictionary is constructed by randomly choosing edge patches from the edge maps of the training set. Shen et al. [2015] construct their edge patch dictionary by clustering edge patches extracted from the training set into K clusters. They employ a deep neural network classifier to choose one the edge patch from the dictionary that most closely matches the content of an input RGB image patch.

Bertasius *et al.* used a Canny edge detector [Canny, 1986] to select candidate contour pixels that are further refined using neural network. The network uses the convolutional layers of the AlexNet ImageNet classification network [Krizhevsky et al., 2012] and predicts two outputs; a classification head predicts if the pixel lies on a contour while a regression head predicts the proportion of human labellers that annotated the pixel as a contour. This is specific to the BSDS-500 dataset [Martin et al., 2001] as it provides multiple ground truths for each image. Xie and Tu [2015] presented a used a fully convolutional architecture that is similar in nature to the FCN architecture of Long et al. [2015a], except that its side taps connect to the convolutional layers of VGG-16 just prior to the max-pooling layers instead of from the max-pooling layers themselves. Contour maps are predicted from the side taps, scaled to full resolution and combined using a learned weighting. Liu et al. [2017] developed an encoder-decoder architecture that uses the convolutional and pooling layers of VGG-16 as a backbone. It's structure bears similarity to that of U-net [Ronneberger et al., 2015].

2.5.3 Instance segmentation

The instance segmentation work that will be discussed here can be divided into two lines of approaches. The first approach consists of using semantic segmentation to perform pixel-wise classification to identify the type of object covering each pixel of the image, after which these contiguous regions are separated into instances. The second approach uses object detection to identify each instance followed by a further step that locates the boundary of each object. Instance segmentation is illustrated in Figure 2.5.

Semantic segmentation and instance separation

The simplest approach to instance segmentation combines semantic segmentation and contour detection [French et al., 2015], using – for example – the Watershed algorithm [Beucher and Meyer, 1993] to split contiguous regions from the semantic segmenter into regions for different object instances, using the output of a contour detector as a guide. A single network can be used to perform both semantic segmentation and contour detection. In practice this approach is often unreliable. The Watershed algorithm uses a flood-fill based approach, so small gaps in the detected contours that are a result of false negatives in the contour predictions result in instances failing to be separated. False positive contour detections can result in the complementary problem of over-segmentation.

Guerrero-Pena et al. [2018] train a U-net [Ronneberger et al., 2015] segmentation network for cell instance segmentation that assigns each pixel in an image one of three labels; background, foreground and touching. During training, the touching class is assigned to pixels that lie on edges separating instances. The class imbalance is addressed using a custom loss weighing scheme.

Bai and Urtasun [2017] developed an approach that predicts an energy map where object instances are represented as energy basins. The energy map is essentially the distance from a pixel to the closest instance edge. They use two networks to achieve this. Their first network predicts a per-pixel direction vector that points away from the closest instance edge. The direction map has sharp discontinuities in which the direction vectors diverge at instance edges and converge at ridges along the centres of object instances. The second network takes the predicted direction map as input and predicts the distance map. The distance map is thresholded to extract contiguous regions representing object instances. Similarly, Uhrig et al. [2016] train a network to predict a discretized angle towards the centre of the object instance along with a depth estimate. Template matching is applied to the predicted angle map to localize object instances.

Dai et al. [2016a] propose InstanceFCN; an extension of fully convolutional networks (FCNs Long et al. [2015a]) that predict instance-sensitive score maps. An instance-sensitive score map predicts if a pixel has one of several discrete relative positions with respect to an instance, e.g. *left side* or *right side*. An instance assembly module locates instances by scanning the score maps in a sliding window fashion.

Object detection and boundary localisation

A parallel branch of research solves the instance segmentation problem in a fashion that is more similar to object detection, predicting the presence of objects along with masks that identify their boundaries.

Pinheiro et al. [2015] proposed DeepMask, a network that takes a patch extracted from an image as input and predicts a segmentation mask along with a score corresponding to the likelihood that the patch is centred on an object. At inference time their network operates in a fully convolutional manner, predicting masks and *objectness* scores a regular intervals across the image. The masks corresponding to high scores are used as object proposals. Based on DeepMask, SharpMask [Pinheiro et al., 2016] uses skip connections to use higher

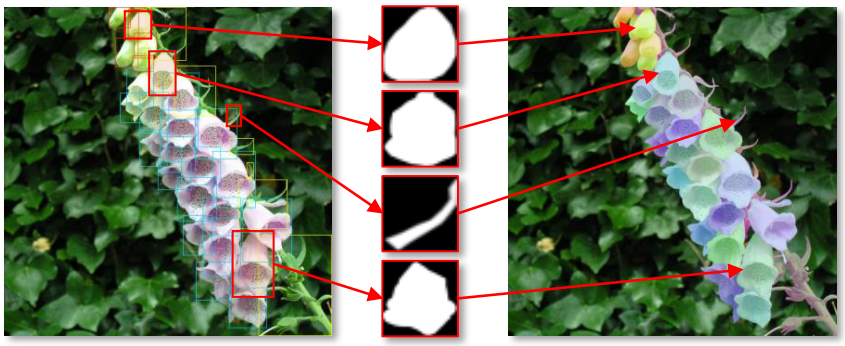


Figure 2.6: Instance segmentation: object detection and boundary localisation, as performed by Mask R-CNN [He et al., 2017]. On the left, four sample detections are highlighted. In the centre the mask predictions for the four detections are shown. On the right the resulting instance segmentation map is overlaid on the input image.

resolution representations from early layers in the VGG-based backbone network to predict higher fidelity masks.

Dai et al. [2016b] presented multi-task network cascades (MNCs) whose structure is similar to that of Faster R-CNN [Ren et al., 2015]. It consists of three stages: a region proposal network (RPN) to propose boxes that likely contain objects; an instance-level mask regressor that predicts a mask for each object proposal; and finally a classifier that categorizes the detected object. As with Faster R-CNN the three stages draw input from a shared backbone network; VGG in the case of MNC. The stage-1 region proposal network (RPN) remains unchanged from Faster R-CNN. Like the R-CNN stage of Faster R-CNN, the stage-2 mask regressor uses RoI pooling to extract a fixed size patch from the feature image generated by the shared backbone. This patch is passed through a network that predicts a 28×28 mask. Finally the stage-3 object classifier takes the feature patch extracted by RoI pooling and 'masks' is using the mask prediction and classifies the object.

He et al. [2017] proposed Mask R-CNN, an enhancement to Faster R-CNN that adds mask regression as a third stage, instead of inserting it between the RPN and the classifier, as in MNC. Mask-RCNN therefore consists of: a stage-1 RPN that generates object proposals; a stage-2 R-CNN network that classifies objects and refines bounding boxes; and finally a stage-3 class specific mask regressor. For each object the mask regressor predicts C masks, where C is the number of classes recognised by the network. This allows the network to learn per-class mask shape priors, improving accuracy. Furthermore Mask-RCNN gains additional accuracy by using a backbone based on residual networks [He et al., 2016] and feature pyramid networks (FPNs) [Lin et al., 2017]. Mask-RCNN is illustrated in Figure 2.6.

In addition to achieving strong object detection results the Swin [Liu et al., 2021] and the improved Swin V2 [Liu et al., 2022] models yielded strong instance segmentation performance with transformer based architectures.

2.6 Data augmentation

Data augmentation is used to reduce over fitting during training by artificially expanding the training set. It is an important component of the training regimes used for many state-of-the-art image classifiers [Krizhevsky et al., 2012; He et al., 2016; Szegedy et al., 2015; Beyer et al., 2022]. Existing samples are modified using ground truth preserving transformations; usually image transformations for computer vision applications. Augmenting each sample multiple times with different stochastic transformation parameters – or randomly on-the-fly during training – can improve the diversity of the training set. It should be noted however that the increase in diversity and performance is limited by the range of transformations available; data augmentation is no substitute for additional real data.

For image classification problems using small image datasets such as CIFAR-10 [Krizhevsky et al., 2009] this normally comprises random crops and horizontal flips. The 32×32 pixel images are padded by 4 pixels on each edge, to a resolution of 40×40, after which a random 32×32 is chosen; it is in effect random translation. The more challenging ImageNet Russakovsky et al. [2015] dataset necessitates a more elaborate cropping scheme. It was first proposed by Szegedy et al. [2015] alongside their *Inception* architecture and has become known as *Inception crop*. A random crop is chosen from the image whose size is randomly chosen such that it covers between 8% and 100% of the image area and such that its aspect ratio varies between 3/4 and 4/3. This crop is extracted from the image and resized to the network input size; 224×224 for most architectures.

2.6.1 Mixing and masking

Cutout – proposed in DeVries and Taylor [2017] – augments an image by masking a randomly chosen rectangular region to zero. The rectangles have a fixed size but are randomly positioned. In effect it acts as a large scale geometric Dropout [Srivastava et al., 2014], encouraging the network to utilise a wider variety of image features by randomly choosing regions of the image to mask out. Cutout yielded significant performance gains in supervised image classification. Cutout uses randomly positioned rectangles with a fixed size. Similarly Zhong et al. [2020] propose RandErase, in which the contents of a randomly chosen rectangle are replaced with noise.

The MixUp approach of Zhang et al. [2018] improves the performance of supervised image, speech and tabular data classifiers by using interpolated samples during training. Pairs of samples – consisting of input images x_a, x_b and target labels y_a, y_b – are randomly chosen, along with corresponding per-pair blending factors p. The images and labels in each pair are blended using the blending factors: $x_m = (1-p)x_a + px_b$ and $y_m = (1-p)y_a + py_b$. x_m and y_m are used for training as normal. Zhang *et al.* found that the blending factors p should be drawn from the β distribution.

CutMix – presented in Yun et al. [2019] – combines aspects of MixUp and CutOut. Instead of mixing samples using a constant per-pair blending factor, they blend input images using a mask M – $x_m = x_a \odot (1-M) + x_b \odot M$ – and blend target labels using p; the proportion of pixels in M having a value of 1. The pixels in M are initialized to 1 and the pixels within a randomly chosen rectangle are set to 0. In effect, a rectangular region from x_a is cut

and pasted over x_b. In contrast to Cutout, CutMix uses a rectangle whose size is randomly chosen such that $p \sim \mathcal{U}(0, 1)$. For supervised classification problems, CutMix was found to outperform Cutout and MixUp.

2.6.2 Rich augmentation

AutoAugment [Cubuk et al., 2019] and the more recent RandAugment [Cubuk et al., 2020] utilize a repertoire of 14 image transformation operations provided by the Pillow [Lundh et al., 2020] library. AutoAugment also uses Cutout [DeVries and Taylor, 2017] and sample pairing [Inoue, 2018].

AutoAugment uses learned augmentation policies. An augmentation policy comprises 5 sub-policies, each of which combines two image augmentation operations, that are applied with a given probability and strength. The probability, strength and choice of operations are optimized to maximize classification performance using reinforcement learning, requiring a large amount of computation to do so.

RandAugment [Cubuk et al., 2020] is somewhat simpler and has two hyper-parameters; the number of image operations to use to augment each sample and a global strength parameter that determines the strength of every operation used. The hyper-parameters are optimized using grid search.

2.7 Semi-supervised learning

While deep neural networks have set state of the art results in many computer vision problems, this comes at the cost of requiring large quantities of manually labelled training data.

Owing to the wide availability of low-cost cameras, large quantities of image data can often be acquired at very low cost. Producing ground truth labels for these images however is often a bottleneck, as it is a laborious, time consuming and expensive process. Semi-supervised learning offers a potential solution to this problem by requiring ground truth labels for only a small subset of the available training data, learning from the remaining unlabelled training samples in an unsupervised fashion. This is a natural fit for many practical computer vision problems, as it offers to reduce the amount of ground truth labels required, along with their associated cost.

Ouali et al. [2020] present a very thorough overview of a variety of semi-supervised learning methods that we would recommend to any reader who wishes to get a good understanding of the field. In this section we will give a broad overview of deep learning based semi-supervised techniques, with a more in-depth focus on consistency regularization based approaches as they are more relevant to the work presented in this dissertation. The techniques presented here apply to image classification scenarios in which a small subset of the training set is used as labelled data, with the remaining samples (or all samples) being used to train the network in an unsupervised fashion.

2.7.1 Auto-encoders

Auto-encoders are unsupervised neural network models that reconstruct their input samples by first encoding them into a latent space and then decoding and reconstructing them.

Rasmus et al. [2015] proposed ladder networks; a variation of auto-encoders in which lateral connections between the encoding and decoding stages incorporate the layer-by-layer reconstruction objectives of multiple stage layer-wise training into a network that can be trained in an end-to-end fashion. With the addition of noise, ladder networks achieve impressive results on the MNIST and CIFAR-10 benchmarks.

Kingma et al. [2014] developed a two model semi-supervised classification algorithm based on variational auto-encoders [Kingma and Welling, 2014]. The first model transforms the image into a Gaussian distributed latent space, while the second is a conditional variational auto-encoder that incorporates class categories. Maaløe et al. [2016] proposed a variational auto-encoder based model that can be trained in an end-to-end fashion and further improves on semi-supervised classification accuracy.

2.7.2 Generative Adversarial Networks

Generative adversarial networks are generative models [Goodfellow et al., 2014a] that learn to generate samples – *e.g.* images – whose distribution matches that of the target dataset in an unsupervised fashion. A GAN is composed of two networks; a discriminator and a generator. The discriminator is trained to classify samples as coming from either the true data set (real) or from the generator (fake). The generator is trained to produce samples that trick the discriminator into classifying them as real by propagating the gradient from the discriminators 'real' prediction through the discriminator into the generator. Radford et al. [2015] demonstrated that the intermediate feature representation produced within a discriminator can be used for classification.

Springenberg [2015] developed a GAN based classification model that can be trained in a semi-supervised or unsupervised fashion. The discriminator operates as an N-way classifier and is trained to minimise the entropy of the predicted class probabilities for real samples and maximise entropy for generated ones. The generator is trained to generate samples that will maximise a the discriminators class probability prediction for a specified class.

Salimans et al. [2016] use an $N+1$-way classifier with the additional class representing *fake* samples. The discriminator is trained to maximise the predicted probability of the ground truth class for labelled samples and the fake class for generated samples. It attempts to maximise the sum of the predicted probabilities for the real classes for unlabelled samples. Salimans *et al.* also introduced two techniques for stabilising GANs; mini-batch discrimination and feature matching. Mini-batch discrimination has the discriminator operate on multiple samples rather than individual samples. This allows it to detect a lack of diversity among generated samples; normally a good indication that the generator could be about to 'collapse' to a state of generating a constant output. Attempting to maximise the 'real' score now requires the generator to incorporate diversity among samples, avoiding this problem. A feature matching GAN uses a generator trained to produce samples that induce latent

features within a later layer within the discriminator that match the latent features induced by real samples.

GANs are trained such that the discriminator is used to guide the generator toward producing samples whose distribution closely approximates that of the target dataset. Dai et al. [2017b] demonstrated that semi-supervised classification performance can be improved by training a *complement generator* to approximate a target distribution that assigns high densities for data points with low densities in the true distribution.

2.7.3 Metric embedding

Hoffer and Ailon [2016] describe a metric embedding based semi-supervised learning algorithm that trains a network to place samples within a euclidean embedding space, such that samples within the same class are closer to one another than to samples from other classes. An unlabelled sample is classified by comparing its distances to labelled samples.

2.7.4 Consistency regularization

Consistency regularization describes a class of techniques in which the network is encouraged to give consistent predictions for unlabelled samples under perturbation or augmentation. These approaches normally combine a standard supervised loss term (e.g. cross-entropy loss) with a consistency loss term. The term consistency regularization was popularized in Oliver et al. [2018]. They provide an overview and evaluation of semi-supervised learning approaches, namely entropy minimization, pseudo-labelling and consistency regularization based approaches. They also contrast them against transfer learning, showing that transfer learning is more effective than any semi-supervised approach on the CIFAR-10 dataset.

In essence, an unsupervised sample x is perturbed to produce $\hat{x}$ and the consistency loss term L_{cons} penalises the network f_θ for producing inconsistent predictions by measuring the distance between them, using some distance measure $d(\cdot, \cdot)$: $L_{cons} = d(f_\theta(x), f_\theta(\hat{x}))$.

Sajjadi et al. [2016b] maintain a history of predictions for each unlabelled sample in the training set. At each epoch their consistency loss term is computed as the sum of the square of the differences between the prediction for a sample given by the network in its current state and the predictions generated for the sample in previous epochs. At each epoch the samples are randomly augmented and different dropout masks are used.

Laine and Aila [2017] present two models; their Π-model and their temporal model. The Π-model passes each unlabelled sample through a classifier twice, using stochastic dropout and data augmentation parameters to provide perturbation. Their unsupervised loss is the mean of the squared difference in class probability predictions resulting from the two presentations of each sample. Their temporal model maintains a per-sample moving average of the historical network predictions and encourages subsequent predictions to be consistent with the average. Their approach achieved state of the art results in the SVHN and CIFAR-10 semi-supervised classification benchmarks. It is worth noting that the model of Sajjadi et al. [2016b] and the temporal model of Laine and Aila [2017] both require per-

sample predictions to be stored for the complete dataset and thus can have a large memory footprint.

The approach of Tarvainen and Valpola [2017] encourages consistency between predictions generated by two neural networks; the student f_θ and the teacher g_ϕ. The student network is trained as normal using gradient descent. The weights of the teacher are updated using Polyak averaging [Polyak and Juditsky, 1992]; they are an exponential moving average of those of the student. The unsupervised loss used to train the student is the mean square difference between the predictions of the student and the teacher, under different dropout, noise and image translation parameters.

Szegedy et al. [2014] discovered that deep neural networks are vulnerable to misclassification of images that are subtly and (almost) imperceptibly modified such that they retain their original visual appearance while causing the network to mis-predict. Goodfellow et al. [2014b] investigated and explained the properties of adversarial examples and used them to provide a form of regularization in order to reduce over-fitting. Miyato et al. [2017] introduced virtual adversarial training (VAT), in which perturbation takes the form of adversarial examples generated to attempt to fool the classifier.

Interpolation Consistency Training (ICT) [Verma et al., 2019] combines MixUp with the Mean Teacher model. Rather than encouraging consistent predictions for an unsupervised sample x and its perturbed variant $\hat{x}$, ICT enforces consistency under mixing, encouraging predictions to vary linearly between pairs unsupervised samples. The teacher network is used to predict class probabilities for a pair of images x_a and x_b and MixUp is used to blend the images $x_m = (1-p)x_a + px_b$ and the teachers' predictions $y_m = (1-p)g_\phi(x_a) + pg_\phi(x_b)$. The predictions of the student for the blended image $f_\theta(x_m)$ are encouraged to be as close as possible to the blended teacher predictions.

MixMatch [Berthelot et al., 2019b] combines standard augmentation based perturbation and MixUp. It stochastically perturbs each sample multiple times and averages the predictions to produce unsupervised targets. MixUp blends labelled samples with corresponding ground truths and unlabelled samples with corresponding unsupervised targets.

Unsupervised data augmentation (UDA) [Xie et al., 2019] perturbs samples with CutOut [DeVries and Taylor, 2017] and RandAugment [Cubuk et al., 2020]; a rich augmentation scheme discussed in Section 2.6.2. They encourage consistency between predictions for unsupervised samples and the same samples perturbed using RandAugment and CutOut.

ReMixMatch [Berthelot et al., 2019a] extends MixMatch, adding distribution alignment and augmentation anchoring along with CTAugment; a new rich augmentation scheme that learns an augmentation policy in the same vein as RandAugment or AutoAugment on the fly. They use two augmentation schemes; a 'weak' scheme consisting of standard operations such as flip and crop and a 'strong' scheme, that combines Cutout with CTAugment. The predictions arising from a weakly augmented sample are used as a pseudo-target for K stochastic strongly augmented presentations of the same sample. Distribution alignment scales the pseudo target so that the running average of the predicted probabilities for unlabelled samples is encouraged to match the distribution of the labelled subset.

The FixMatch model of Sohn et al. [2020] is one of the simplest semi-supervised classification models to date while being significantly simpler than the prior art it improves on. Like ReMixMatch it uses two augmentation schemes; 'weak' augmentation consisting of standard operations such as flip and crop and 'strong' augmentation combining Cutout with RandAugment or CTAugment. Pseudo labels predicted for 'weakly' augmented samples are used as training targets (using cross-entropy loss) for 'strongly' augmented variants of the same samples.

The success of UDA, ReMixMatch and FixMatch demonstrate the benefit of rich data augmentation, as provided by CutOut [DeVries and Taylor, 2017] and RandAugment [Cubuk et al., 2020] (UDA) or CTAugment [Berthelot et al., 2019a] (FixMatch). We also note that many approaches introduced in the last few years predict pseudo-targets for unsupervised samples. Both VAT [Miyato et al., 2017] and UDA [Xie et al., 2019] stop gradient propagation through the prediction $f_\theta(x)$ from the original unlabelled sample x, using it as a target for the perturbed $f_\theta(\hat{x})$. The Mean teacher model [Tarvainen and Valpola, 2017] encourages similarity between the predictions of its teacher and student networks. Given that the teacher weights are an exponential moving average of those of the student, mean teacher also does not propagate gradients though the teacher branch of the consistency loss term, using the teacher predictions as targets for training the student. FixMatch of course explicitly uses label propagation, predicting a hard one-hot label used as a training target.

CoMatch [Li et al., 2021] combines consistency regularization with self-supervised contrastive learning (discussed below in Section 2.9.2). Sample similarity between contrastive embeddings computed for unsupervised samples are used to compute weighted average pseudo-labels, thereby using similarity to other samples to improve the quality of the pseudo-label used as an unsupervised training target. Furthermore, agreement between a pseudo-label graph and a contrastive embedding similarity graph encourages clustering.

2.7.5 Other approaches

Meta Pseudo Labels [Pham et al., 2021] combines pseudo labelling – in which a teacher network predicts labels used to train a student – with meta-learning objectives that ensure that use the performance of the student on supervised samples to guide the training of the teacher.

2.7.6 Semi-supervised semantic segmentation

A number of approaches for semi-supervised semantic segmentation use additional data. Kalluri et al. [2018] use data from two datasets from different domains, maximizing the similarity between per-class embeddings from each dataset. Stekovic et al. [2018] use depth images and enforced geometric constraints between multiple views of a 3D scene.

Prior to our work discussed in Chapter5 that finished early in 2020 relatively few approaches operated in a strictly semi-supervised setting. Hung et al. [2018] and Mittal et al. [2019a] employ GAN-based adversarial learning, training a standard semantic segmentation model in the place of the GAN generator and using a discriminator to distinguish ground truth from

predicted segmentation maps. For unsupervised samples the segmentation model is trained to fool the discriminator by producing more realistic segmentation maps. The discriminator developed by Hung *et al.* has a small receptive field and is applied in a sliding window fashion, producing a *real/fake* map as output. In contrast the discriminator developed by Mittal *et al.* takes a channel-wise concatenation of the RGB image and segmentation map as input and produces a *real/fake* score for the complete image. In addition, Mittal et al. [2019a] train a mean teacher model [Tarvainen and Valpola, 2017] to predict the presence or absence of objects of each class in the image and use these predictions to suppress classes in predicted segmentation maps belonging to objects not predicted to be present, further improving performance.

Prior to 2020, the only successful applications of consistency regularisation to segmentation that we were aware of come from the medical imaging community; Perone and Cohen-Adad [2018] and Li et al. [2018b] apply consistency regularization to an MRI volume dataset and to skin lesions respectively using standard augmentation to provide perturbation. Cui et al. [2019] use the mean teacher model [Tarvainen and Valpola, 2017] to segment brain lesion MRI images and perturb unsupervised images with noise.

It is worth noting the challenges involved in using standard augmentation to drive consistency regularisation for semantic segmentation. These augmentation schemes usually involve geometric augmentations such as translation, scaling and rotation. Applying a geometric transformation to an input image will result in the corresponding segmentation map being transformed in the same manner. This needs to be accounted for when computing the consistency loss. This has been addressed by Perone and Cohen-Adad [2018], Li et al. [2018b] and Ji et al. [2019]. We will discuss this in more detail in Section 5.4.1.

2.8 Unsupervised domain adaptation

A neural network based computer vision model can attain excellent performance when applied to data that is similar to that on which it was trained. Applying the model to images with a different appearance to that of the training set (*e.g.* due to difference lighting or capture conditions) often causes the performance to drop. For example, an urban street scene segmentation model trained on daylight images captured in sunny weather will exhibit impaired performance when applied to images captured in cloudy or rainy weather, or at night time. This difference in appearance or distribution between data sources is known as the *domain gap*. Domain adaptation attempts to bridge the domain gap, improving performance on samples drawn from a target domain, that is distinct from the source domain. Typically ground truth labels are relatively plentiful for source domain samples, but few or no ground truth labels are available for the target domain. A strong motivation for the development of unsupervised domain adaptation algorithms is the potential to train using abundantly labelled synthetic data, while achieving good performance on sparsely labelled or unlabelled real-world data.

Unsupervised domain adaptation is a problem closely related to that of semi-supervised learning in which one attempts to transfer knowledge gained from a labelled source dataset to a distinct unlabelled target dataset, with the constraint that the objective (*e.g.* digit

classification) must remain the same. While the labelled and unlabelled data in semi-supervised learning problems must come from the same distribution, in domain adaptation problems the distributions are different.

2.8.1 Auto-encoders

Ghifary et al. [2016] developed an auto-encoder model that is trained to reconstruct samples from both the source and target domains. A classifier is trained to predict labels from domain invariant features present in the latent representation generated by the encoder using source domain labels as ground truth.

Bousmalis et al. [2016] recognized that samples from disparate domains have distinct domain specific characteristics that must be represented in the latent representation to support effective reconstruction. They developed a split model that separates the latent representation into shared domain invariant features and private features specific to the source and target domains. Their decoder uses both the shared invariant features and the private features for reconstruction. Their utilises the domain invariant features only.

2.8.2 Domain adversarial training

Ganin and Lempitsky [2015] propose a bifurcated classifier that predicts the target label of a given sample and the domain (source/target) from which it originated. The classifier splits into label classification and domain classification branches after common feature extraction layers. A gradient reversal layer is placed between the common feature extraction layers and the domain classification branch; while the domain classification layers attempt to learn to recognise features that discriminate between domains, the gradient reversal operation encourages the feature extraction layers to confuse the domain classifier by extracting domain invariant features. An alternative and simpler implementation described in their appendix minimises the target label cross-entropy loss in the shared feature and label classification layers, minimises the domain cross-entropy in the domain classification layers but *maximises* it in the shared feature layers. The model of Tzeng et al. [2017] runs along similar lines but uses separate feature extraction sub-networks for source and target domain samples and train the model in two distinct stages.

Wang et al. [2018] discuss stability problems that can be encountered when using domain adversarial training, noting the similarity with stability problems countered when training GANs. Following solutions developed for stabilizing GANS, they treat the domain classifier as the critic from a Wasserstein GAN [Martin Arjovsky and Bottou, 2017] and employ the gradient penalty (WGAN-GP) approach of Gulrajani et al. [2017] to stabilize domain adversarial training.

2.8.3 Tri-training

Saito et al. [2017a] use tri-training [Zhou and Li, 2005]; feature extraction layers are used to drive three classifier sub-networks. The first two are trained on samples from the source domain, while a weight similarity penalty encourages them to learn different weights. Pseudo-

labels generated for target domain samples by these source domain classifiers are used to train the final classifier to operate on the target domain.

2.8.4 Generative Adversarial Networks

Generative Adversarial Networks based approaches show promise for domain adaptation. Some GAN based models – such as that of Sankaranarayanan et al. [2017] – use a GAN to help learn a domain invariant embedding for samples. Many GAN based domain adaptation approaches use a generator that transforms samples from one domain to another.

Bousmalis et al. [2017] propose a GAN that adapts synthetic images to better match the characteristics of real images. Their generator takes a synthetic image and noise vector as input and produces an adapted image. They train a classifier to predict annotations for source and adapted samples alongside the GAN, while encouraging the generator to preserve aspects of the image important for annotation. The model of Shrivastava et al. [2017] consists of a refiner network (in the place of a generator) and discriminator that have a limited receptive field, limiting their model to making local changes while preserving ground truth annotations. The use of refined simulated images with corresponding ground truths resulted in improved performance in gaze and hand pose estimation.

Russo et al. [2018] present a bi-directional GAN composed of two generators that transform samples from the source to the target domain and vice versa. They transform labelled source samples to the target domain using one generator and back to the source domain with the other and encourage the network to learn label class consistency. This work bears similarities to CycleGAN [Zhu et al., 2017].

2.8.5 Distribution matching

A number of domain adaptation models maximise domain confusion by minimising the difference between the distributions of features extracted from source and target domains. Deep CORAL [Sun and Saenko, 2016] minimises the difference between the feature covariance matrices for a mini-batch of samples from the source and target domains. Tzeng et al. [2014] and Long et al. [2015b] minimise the Maximum Mean Discrepancy metric [Gretton et al., 2012]. Li et al. [2016] described *adaptive batch normalization*, a variant of batch normalization [Ioffe and Szegedy, 2015] that learns separate batch normalization statistics for the source and target domains in a two-pass process, establishing new state-of-the-art results. In the first pass standard supervised learning is used to train a classifier for samples from the source domain. In the second pass, normalization statistics for target domain samples are computed for each batch normalization layer in the network, without modifying the network weights.

2.8.6 Consistency regularization

Shu et al. [2018] describe a consistency regularization based approach to domain adaptation. Their main approach VADA (Virtual Adversarial Domain Adaptation) combines domain adversarial training [Ganin and Lempitsky, 2015] and virtual adversarial training [Miyato et al., 2017]. Domain adversarial training causes the network to learn to minimise the

difference between the distributions of high level features extracted from the source and target domains, while VAT adds a penalty that punishes violation of the cluster assumption. The use of VAT refines the decision boundaries so that they lie in low density regions separating uniformly classed clusters.

2.9 Unsupervised and self-supervised learning

Unsupervised ans self-supervised learning algorithms attempt to train a model without the use of ground truth labels. Learning from unlabelled data is an attractive possibility given the potential reduction in annotation costs, hence it has been well studied by the machine learning community. Self-supervised learning algorithms use an ancillary task for which training targets can be generated automatically. Zhai et al. [2019] train a model for the ancillary task of predicting the amount of rotation applied to an image, achieving impressive semi-supervised learning results on the ImageNet dataset.

2.9.1 Unsupervised clustering

The approach of Hu et al. [2017] enforce consistent cluster assignment under stochastic augmentation. Their consistency loss term is combined with conditional entropy minimization to encourage unambiguous cluster assignment, balanced with marginal entropy maximization to ensure that samples are distributed between the clusters. The former encourages the network to assign samples to clusters while the latter prevents the obvious degenerate solution that places all samples in a single cluster. The invariant information clustering (IIC) loss term of Ji et al. [2019] combines the aforementioned consistency and entropy terms into a single loss term. They successfully apply their approach to classification and semantic segmentation problems.

2.9.2 Self-supervised metric learning

The approaches discussed here train a network to produce an embedding vector f_i given an input image x_i, such that the similarity between embedding vectors is representative of the similarity of the image content.

Triplet loss

Early approaches employed triplet loss [Chechik et al., 2010]; FaceNet [Schroff et al., 2015] is one such example. During training, mini-batches are composed of triplets, each of which consists of an anchor sample x_i^a, a positive sample x_i^p whose identity matches that of the anchor and a negative sample x_i^n with a different identity. The triplet loss term $\mathcal{L}_{triplet}$ encourages the network to generate embeddings such that the positive embedding f_i^p is closer to the anchor f_i^a than the negative embedding f_i^n is to f_i^a by a margin α:

$$\mathcal{L}_{triplet} = \max(\left\| f_i^p - f_i^a \right\|_2^2 + \alpha - \| f_i^n - f_i^a \|_2^2, 0)$$

The authors of FaceNet note that it is crucial to select triplets that maximize the triplet loss for fast convergence. They achieve this using hard negative mining, by selecting f_i^n to be the closest negative sample to the anchor f_i^a.

InfoNCE, NT-Xent and MoCo

Self-supervised approaches based on InfoNCE loss – first presented in Oord et al. [2018] – gained favour among researchers due to their strong performance when used to train image classification networks. Performance is measured using the *linear classification protocol* [He et al., 2020], often using the ImageNet dataset. The linear classification protocol uses transfer learning; it attaches a linear classification layer to a standard image classifier backbone (*e.g.* ResNet-50 [He et al., 2016]) that was pre-trained using a self-supervised learning algorithm. The linear classifier is trained in a supervised fashion using *frozen* features; the layers of the backbone network are left un-modified. Strong classification performance indicates that the self-supervised algorithm trained the network to generated high quality features in which the target classes are linearly separable. Modern self-supervised approaches nearly match the performance of supervised learning using the ImageNet dataset [Chen et al., 2020b,a].

In comparison to triplet loss, modern InfoNCE based approaches compose a mini-batch of anchor-positive pairs x_i and $\hat{x}_i$ and pair the anchor x_i with the set of negative samples $\hat{x}_j | j \neq i$. Furthermore, the loss term uses a similarity measure between the embeddings as pre-softmax logits, with cross-entropy loss maximizing the logit corresponding to the similarity between x_i and $\hat{x}_i$, while minimizing the logits corresponding to the similarity between x_i and $\hat{x}_j | j \neq i$.

When training an image classifier, a mini-batch of N sample images x_i is selected (where $i \in [1, N]$) and a pairs of views v_i and $\hat{v}_i$ are generated for each image. Oord et al. [2018] select a crop paired with nearby context while He et al. [2020] and Chen et al. [2020a] generate views using stochastic augmentation. More recent approaches adopt the NT-Xent [Chen et al., 2020a] definition of the loss term presented in Wu et al. [2018]. A fully-connected projection layer is added to a standard classification network backbone, generating high-dimensional (e.g. 128-d) embeddings f_i and $\hat{f}_i$ given v_i and $\hat{v}_i$ as input. f_i and $\hat{f}_i$ are subsequently normalized to length 1 (projecting them onto the surface of a unit hypersphere), after which embedding similarity is computed using cosine similarity. The NT-Xent loss $\mathcal{L}_{NTXent}$ is computed as:

$$\mathcal{L}_{NTXent} = -\frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{\exp(f_i \cdot \hat{f}_i / \tau)}{\sum_{j=1}^{N} \exp(f_i \cdot \hat{f}_j / \tau)} \right)$$

where τ is the temperature.

In effect, the pairwise similarities between every f_i and every $\hat{f}_j$ are computed, divided by the temperature τ and used as pre-softmax logits. Cross entropy loss maximises similarity between embeddings f_i and $\hat{f}_i$ arising from views of the same image x_i (positive pairs) while contrastively minimizing similarity between f_i and all $\hat{f}_j$ for which $j \neq i$ (negative pairs).

The temperature τ applies a scale factor to the pre-softmax logits; a lower temperature results in softmax producing a sharper probability distribution. This and other aspects of the design of contrastive loss terms are discussed in Chen et al. [2021].

In comparison to triplet loss, InfoNCE benefits significantly from being able to contrast an anchor a sample against a large number of negative samples, instead of only one. This also mitigates the benefits of hard negative mining. The benefits of a large number of negative samples have been confirmed by Chen et al. [2020a] in which they find that larger mini-batches – that provide more negative samples per anchor – result in faster convergence, and by He et al. [2020] where they store the embeddings of negative samples in a dictionary, finding that larger dictionaries lead to better performance.

The larger mini-batch sizes explored in Chen et al. [2020a] come at the cost of increased GPU memory requirements, often requiring the use of many GPUs in parallel in order to provide sufficient memory. Wu et al. [2018] alleviated this through the use of a memory bank that stores an embedding for every sample in the dataset. After each mini-batch, the predicted embeddings are inserted into the memory bank. This approach has two drawbacks, namely that storing embeddings for huge datasets (*e.g.* on the order of billions of samples) is infeasible and that the memory bank will contain out of date embeddings that were generated by a network that has changed significantly since those samples were last process due to a large number of training iterations having taken place in the intervening time.

The drawbacks arising from the use of a memory bank are addressed by the MoCo model of He et al. [2020]. MoCo maintains a dictionary of embeddings for the K most recently seen samples (*e.g.* using $K = 65536$ for the ImageNet dataset), placing an upper bound on memory requirements. Furthermore, the embeddings stored in the dictionary are generated by a *key* network, whose weights are computed as an exponential moving average of those of the *query* network that is trained using gradient descent as normal.

We would now like to note the similar approaches adopted by self-supervised learning and semi-supervised learning algorithms. The memory bank used by Wu et al. [2018] bears significant resemblance to the temporal ensembling model of Laine and Aila [2017], that uses a memory bank of predicted class probability vectors. The weights of the *key* network of the MoCo model [He et al., 2020] are updated using Polyak averaging [Polyak and Juditsky, 1992], just as the teacher network uses Polyak averaging in the Mean Teacher model Tarvainen and Valpola [2017].

Chen et al. [2020a] explored the design of the projection head that is attached to the network backbone. The function of the projection head is to predict an embedding vector for a given input image. While prior work [Wu et al., 2018; He et al., 2020] used a single fully-connected layer, Chen et al. [2020a] found that a two-layer MLP improved performance. This was adopted by a recent improvement to the MoCo model [Chen et al., 2020b], closing the gap between self-supervised and supervised learning.

Augmentation for self-supervised learning

Wu et al. [2018] and He et al. [2020] used data augmentation to stochastically create views of a sample image. Chen et al. [2020a] conducted a rigorous ablation study in which they analysed the effectiveness of different forms of image augmentation. They found that applying a colour jitter augmentation is essential to good performance, as otherwise the network can rely on image colour statistics as a short-cut for comparing image identities. Using colour jitter requires the network to focus on higher level features that represent the image content. They also found Gaussian blur augmentation to be helpful. This was also adopted by MoCo V2 [Chen et al., 2020b].

2.10 Active learning

Active learning aims to reduce the manual annotation burden by using a machine learning model to prioritise unlabelled samples for annotation. It rests on the premise that annotating some samples achieves a greater gain in performance than annotating others; not all samples contribute equally to the performance of the model. By prioritising the samples in order to decreasing performance gain, the most useful samples are annotated first, maximizing the value obtained per unit of manual annotation labour.

The effect of active learning for deep learning models was explored by Wang and Shang [2014], with a focus on MNIST digit classifiers. A classifier model is trained on a small seed annotated subset. The labelled subset is grown by selecting un-annotated samples for labelling using three metrics based on the probability predictions from the model. The least confidence metric computes the predicted confidences (probability of predicted class) of all un-annotated samples and selects those with the least confidence. Margin sampling selects the samples with the smallest separation between the top two class predictions. Lastly the entropy metric is used to select samples whose predictions have the highest entropy. These metrics exhibit similar performance and will in effect select samples that are closest to the decision boundary.

The study of Mittal et al. [2019b] suggests that current active learning methods have significant limitations. When contrasted to choosing samples for annotation in a random order, current active learning approaches yield performance benefits in plain supervised learning scenarios. These approaches are however significantly out-performed by semi-supervised learning algorithms for a given annotation budget. Furthermore, using them in concert with semi-supervised learning causes the effect of active learning methods to diminish to the point at which they offer little if any benefit over a random sample order.

2.11 Feature extraction and matching for panorama creation

A panorama is created by stitching a number of partially overlapping images together, aligning the overlapping regions such that a large contiguous image is formed. The most common approach starts by detecting key-points within each image, and extracting a feature descriptor for each key-point. The feature descriptors are then matched – usually by closest distance – and the key-point matches between a pair of images are used to estimate the

fundamental matrix in order to align the images. While the goal of by-catch quantification has little use for panorama creation, the process can be used to estimate the motion of the conveyor belt, as discussed in Section 7.4.3.

The first step of the process involves selecting key-points in each image such that the content of the surrounding region is distinct, thereby increasing the chance of good quality matches. Corners are usually chosen and smooth, featureless regions of the image are be avoided. A number of approaches to corner detection have been proposed including Harris corner detectors [Harris et al., 1988], difference of Gaussians [Lowe, 2004] and the FAST detector [Rosten and Drummond, 2006].

The second step is feature extraction, in which a feature descriptor is generated for each key-point that describes the content of the surrounding region. Effective feature detectors such as SIFT [Lowe, 1999] and SURF [Bay et al., 2006] generate a descriptor that is invariant to scale and orientation, allowing key-point matching to be robust to variation in pose within the source images. The BRIEF descriptor [Calonder et al., 2010] is a binary feature descriptor that is often as effective as SURF but requiring considerably less computation. ORB [Rublee et al., 2011] combines an oriented variant of FAST [Rosten and Drummond, 2006] with a rotation-aware variant of BRIEF, resulting in a fast and effective system for key-point detection and feature extraction.

Finally, key-points residing within the overlapping regions of a pair of images are matched and the fundamental matrix is estimated. Descriptors from image A are greedily matched with descriptors from image B, without at first considering the location of the corresponding key-points. This normally results in a large number of matches, many of which are spurious and would result in a poor estimate of the fundamental matrix. The RANSAC algorithm [Fischler and Bolles, 1981] is used to reject spurious matches that are outliers with respect to the perspective projection model that is the fundamental matrix.

2.12 Object tracking

An object tracker should detect an object instance in a frame of video and follow the object throughout subsequent frames while it remains visible. Multiple object tracking concerns detecting and tracking all objects of interest, re-identifying object instances when they become visible after going out of view temporarily, perhaps due to occlusion. When an object goes out of view temporarily, its track can be said to be split into a number of tracklets. A tracklet is a sequence of contiguous frames and corresponding detections belonging to an object. Some approaches use an object re-identification model to match tracklets that correspond to the same object, allowing objects to be tracked in spite of temporary occlusions.

A common approach is to use a discriminative appearance model that consists of a classifier that predicts the presence or absence of the tracking target in an image patch. Kernelized Correlation Filters (KCF) by Henriques et al. [2014] leverage cyclic shifts and the Fourier domain to quickly train a classifier on many positive and negative patches in a cyclic sliding window fashion. The result is a very fast and effective approach for real-time object tracking. While deep learning methods for object tracking have been proposed [Valmadre

et al., 2017], the use of neural networks imposes significant computational and pre-training requirements.

The SORT (Simple On-line and Real-time Tracking) algorithm of Bewley et al. [2016] joins per-frame detections generated by a Faster R-CNN object detector into tracklets. Tracked objects – or targets – are represented as moving bounding boxes; their state consists of the object position, area and aspect ratio, along with positional and size velocities. The velocities are used to estimate the bounding box of a target in a subsequent frame, after which these estimates are matched to the detections produced by the object detector using the Hungarian algorithm [Kuhn, 1955]. A Kalman filter [Kalman, 1960] is used to update the state of a tracked object when a new detection is associated with it.

Wojke et al. [2017] presented DeepSORT, extending SORT with a deep network model trained to extract appearance features for the purpose of re-identification. The appearance features of tracklets are computed and retained, so that should a missing object (e.g. due to occlusion) be re-detected, the appearance features can be used to re-identify it and associate it with a prior missing tracklet.

The Tracktor approach of Bergmann et al. [2019] re-uses the bounding box regressor of the R-CNN head of a Faster R-CNN object detector network for tracking. Detections from frame $t-1$ are carried forward to frame t, after which the bounding box regression component of the R-CNN head refines the bounding boxes, fitting them to the new positions of the objects. Furthermore, the classification component determines if the detections should be retained or discarded due to going out of view. The re-use of an existing object detection network eliminates the need for training a tracking specific model; a standard object detection model is all that is required.

The CenterTrack approach of Zhou et al. [2020] trains an object detection and tracking network that takes the current and previous frame as input along with a heat-map that provides the centre points of detections in the previous frame. It is an extension of CenterNet [Zhou et al., 2019] that uses a similar approach for single frame object detection. It predicts a centre point heat-map for the current frame, corresponding bounding box sizes and offsets that provide a predicted motion vector. The offset vectors are used to associate detections between frames, propagating object identity.

Karthik et al. [2020] present an unsupervised method for training an object re-identification model. They apply a pre-trained object detection network frame-by-frame to detect objects of interest in video segments and apply the SORT algorithm to join them into noisy tracklets. The tracklets are used to train an object re-identification model to associate presentations of an object throughout different frames with one another, while discriminating between different objects. Af inference time, they employ the DeepSORT approach; they use a CenterNet object detection model to provide frame-by-frame detections and the re-identification model to join tracklets.

We note that both DeepSORT and CenterTrack require an object tracking dataset with ground truth tracks. CenterTrack uses pairs of subsequent frames and corresponding tracks to create ground truth offsets for training the detection and tracking model. DeepSORT

extracts different views of an object from different frames to train the appearance extraction model to extract features relevant for re-identification.

The object motion apparent in videos arises from two sources; the non-rigid motion of the tracked objects within the scene and the rigid motion of the camera (we adopt the parlance of Han et al. [2020]). Choi and Savarese [2010] and Wang et al. [2019a] improve the prediction capabilities of their motion model by estimating the motion of the camera in addition to the motion of the tracked objects.

2.13 Computer vision for fish classification

The first attempts to apply computer vision to the problem of fish classification were reported in the 1980s by Tayama et al. [1982], who used shape descriptors derived from binary silhouettes to discriminate between 9 fish species with 90% accuracy. Further work combined colour and shape descriptors [Strachan, 1993] achieving a reliability of 100% and 98% in identifying 23 species under laboratory conditions. It involved a mechanical feeding system to ensure that individual fish are correctly oriented and presented to the camera one-by-one, along with tightly controlled lighting. The author notes potential caveats due to seasonal changes in the physical condition of fish and variability in the colour of individual specimens, depending to some extent on the area in which they are caught.

Further work refined approaches for fish species classification using primarily shape and colour features with fuzzy classifiers and neural networks [Hu et al., 1998; Storbeck and Daan, 2001; Alsmadi et al., 2009]. White et al. [2006] describe trials of CatchMeter; a sorting machine capable of measuring and classifying fish based on colour and shape features that achieves fish length measurement accuracy of $\sigma = 1,2$mm and species classification accuracy of flat- and round-fish of approximately 99%. Specimens must be presented individually, but can be in any orientation.

Later research investigates colour, shape and texture features and more advanced classifiers but still requiring constrained environments avoiding occlusion. As a consequence, counting individuals is trivial or irrelevant (e.g. Hu et al. [2012]). However, a review of computer vision in aquaculture and processing of fish products identifies a wide range of applications for the technology at all stages of production [Mathiassen et al., 2011; Zion, 2012], many of which present challenging problems for computer vision.

Successfully classifying images captured in real-life conditions requires the use of more sophisticated approaches such as non-rigid part models [Chuang et al., 2016]. Deep neural network based feature extractors have been successfully employed for fish species identification on the Fish4Knowledge [Boom et al., 2012], using unsupervised learning to initialise the network layers [Sun et al., 2016; Qin et al., 2016]. More recent work employs deep neural network image classifiers trained in an end-to-end fashion [Zheng et al., 2018], tackling a challenging Kaggle dataset [The Nature Conservancy, 2016] in which equipment and personnel are present in the images, in addition to the fish.

2.14 Discussion

We have discussed the prior work that is relevant to the research that we present in the following chapters in this thesis. We have learned about the structure of object detection and instance segmentation models that we utilize in Chapters 3 and 7. The understanding of prior work that we gained in the areas of unsupervised domain adaptation and semi-supervised learning was essential in informing the directions in which we took our research in Chapters 4, 5 and 6.

3 Multi-spectral object detection

Multi-spectral images typically include channels beyond the normal RGB that represent parts of the electromagnetic spectrum outside the human visual range, such as infrared or ultraviolet. Multi-spectral images captures by satellites are used for weather forecasting [Tatem et al., 2008] and object detection [Bowler et al., 2019]. Thermal imagery utilizes infra-red bands to quantify the temperature of objects within a scene. This is often used in night-time surveillance applications as no external illumination is required.

In this chapter (first presented in our paper French et al. [2018a]) we focus on object detection – specifically pedestrian detection – using multi-spectral images. We contribute an exploration of the use of the spectral edge image fusion [Connah et al., 2015] algorithm to fuse the thermal band of a multi-spectral image into the RGB band as a pre-process, prior to use as an input to an object detection neural network. The use of image fusion permits the use of a standard pre-trained RGB object detection network without requiring the architectural modifications that are required to handle multi-spectral input. While object detection networks that accept multi-spectral input achieve higher performance, we demonstrate that fused RGB images produced with spectral edge image fusion retain useful visual cues from the thermal band, helping the network achieve higher performance than can be achieved using plain RGB images. The use of image fusion also highlighted a disparity between the RGB and thermal bands that is present in the KAIST [Hwang et al., 2015] pedestrian detection data set.

3.1 Pedestrian detection

Accurate pedestrian detection is an essential component of autonomous driving and surveillance systems and is a specific and challenging use case of general object detection. The appearance of pedestrians in daylight varies considerably due to a differences in clothing, pose and distance from the camera. Night time conditions present a particularly challenging scenario as low light levels can render pedestrians nearly indistinguishable from the image background.

The work of Hwang et al. [2015] demonstrates that multi-spectral imagery consisting of RGB and long-wave infra-red thermal images improves the accuracy of pedestrian detection systems as the thermal images contain additional visual cues that a pedestrian detection system can utilise. The benefits of multi-spectral imagery are particularly apparent at night time, as poor lighting conditions have far less effect in long-wave infra-red bands.

Multi-spectral pedestrian detection systems developed thus far use either a tetra-chromatic input that combines RGB and thermal channels, or perform feature extraction on the visible and thermal bands separately and combine features drawn from RGB and infra-red bands later in the pipeline. Neural network based pedestrian detection systems utilise similar

structures. Jingjing Liu and Metaxas [2016] developed a Faster-RCNN neural network based object detection network that accepts multi-spectral (RGB and thermal) input and confirm the clear advantages of using multi-spectral imagery over RGB only.

3.2 Image fusion

Image fusion is a process in which the channels in a multi-channel image (the channels can differ in resolution in some situations) are combined, reducing the number of channels. Examples include colour to grey-scale conversion and converting multi-band satellite imagery to RGB images. An effective image fusion algorithm attempts to preserve as much salient visual information from the original high-channel image during the fusion process.

3.2.1 Spectral edge fusion

In this chapter we use the spectral edge image fusion algorithm of Connah et al. [2015]. It is an approach for transforming an N-channel image to an M-channel image where $N > M$ while preserving contrast and gradient information such that the lower dimensional image can preserve RGB appearance while incorporating gradient and contrast information from the other channels. Frequent use cases involve enhancing an RGB image with information from other channels within a multi-spectral image, such that details from invisible parts of the spectrum can be perceived by the viewer, while retaining much of the original colour of the RGB image so that it can be easily interpreted. The approach is outlined in Figure 3.1.

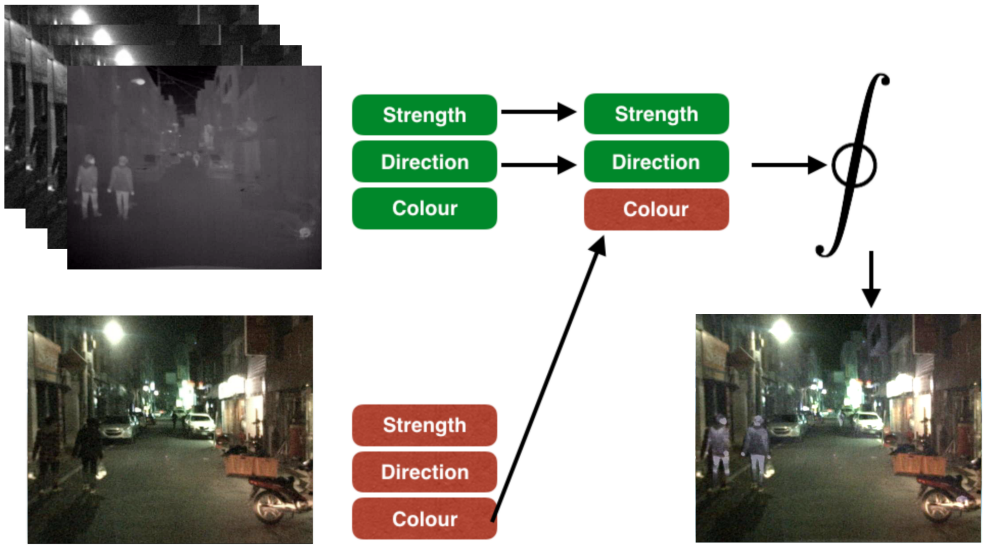


Figure 3.1: An example of Spectral Edge Image Fusion being used to fuse RGB+NIR to make a new RGB image

In Figure 3.1 top left we show a 4 channel image split into 4 greyscale single channels. The RGB channels are drawn from the picture shown in the bottom left. The front grey-scale

image shows the Near Infra-red information. The RGB channels are used as a 'guide' image that provides the base colour information used to construct the resulting image.

The algorithm in detail

We will now describe the spectral edge fusion algorithm in more detail. Much of this material was drawn from Connah et al. [2015].

We define the multi-channel gradient ∇C for a single pixel as follows, which we compute by local finite differencing:

$$\nabla C = \begin{bmatrix} C^1_{,x} & C^1_{,y} \\ \vdots & \vdots \\ C^N_{,x} & C^N_{,y} \end{bmatrix}$$

The vector $[C^i_{,x}, C^i_{,y}]$ represents the gradient of the i-th channel at the pixel. We can project the gradient onto a direction vector $d = [cos\theta, sin\theta]^T$ to obtain the gradient magnitude in d by computing ∇Cd. The squared magnitude of the gradient in d is $m^2 = d^T(\nabla C)^T \nabla Cd$.

The 2×2 structure tensor $Z_C = (\nabla C)^T \nabla C$ encodes magnitude information of the gradient in two dimensions, therefore representing the directional contrast. It can be used to obtain the squared gradient magnitude in any direction d by computing $d^T Z_C d$. Written in full Z_C is:

$$Z_C = \begin{pmatrix} \sum_k C^k_{,x} C^k_{,x} & \sum_k C^k_{,x} C^k_{,y} \\ \sum_k C^k_{,x} C^k_{,y} & \sum_k C^k_{,y} C^k_{,y} \end{pmatrix}$$

The Spectral edge algorithm constrains the contrast of the resulting M-channel image gradient to be the same as that of the original N-channel gradient image by ensuring that the structure tensor for both is identical. If we represent the $N \times 2$ gradient matrix ∇C as its singular value decomposition $U\Lambda V^T$, we decompose ∇C into the $N \times 2$ colour matrix U, spatial information V and magnitude Λ (*colour*, *direction* and *strength* in Figure 3.1). We can obtain the structure tensor Z_C as:

$$Z_C = (\nabla C)^T \nabla C = V \Lambda^T U^T U \Lambda V^T = V \Lambda^2 V^T$$

as U is orthonormal and $\Lambda = \Lambda^T$. The structure tensor therefore depends only on the spatial information V and the magnitude Λ.

Given an N-channel input image H, an M-channel guide image R – e.g. an RGB image – we compute a re-integrated M-channel image $\tilde{R}$. The structure tensors are defined as:

$$Z_H = (\nabla H)^T \nabla H$$

$$Z_R = (\nabla R)^T \nabla R$$
$$\tilde{Z_R} = (\widetilde{\nabla R})^T \widetilde{\nabla R}$$

We create a new gradient image $\widetilde{\nabla R}$ whose structure tensor matches that of H by combining the SVD components of H and R as follows:

$$\widetilde{\nabla R} = \tilde{U}_R \tilde{\Lambda}_H V_H^T$$

The gradient image $\widetilde{\nabla R}$ is re-integrated integrated to form the final image $\tilde{R}$ – Figure 3.1 lower right – using a lookup-table based approach Finlayson et al. [2011].

The example in Figure 3.1 illustrates that the bright appearance of the pedestrians that can be seen in the NIR image has been incorporated into the final image.

3.3 Method

3.3.1 Network structure

Following prior approaches for pedestrian detection [Jingjing Liu and Metaxas, 2016; Zhang et al., 2016; König et al., 2017], we employ transfer learning, utilising the VGG-16 [Simonyan and Zisserman, 2014] image classification network as the feature extraction backbone.

Similar to the work of König et al. [2017] we base our pedestrian detection system on region proposal networks (RPN); the first stage of a Faster-RCNN object detector. We note the similarity between our RPN-based model and that of a single-shot detector (SSD) [Liu et al., 2016].

Faster R-CNN based object detection systems typically use three anchor box aspect ratios; 1:2, 1:1 and 2:1 (height:width) and four anchor box scales (32×32, 64×64, 128×128, and 256×256 pixels). Following Jingjing Liu and Metaxas [2016] we discarded the 1:2 aspect ratio given the typical aspect ratio of pedestrian bounding boxes, but retained the four different scales.

3.3.2 Protocol

We followed the protocol of Jingjing Liu and Metaxas [2016], ignoring ground truth pedestrian instances that are marked as occluded or containing multiple pedestrians, or whose height is < 50 pixels. Ignored instances are prevented from contributing to the neural network's loss function during training, thus the network is not trained to predict them as being either pedestrians or background. They also do not count during evaluation. Only frames that contain at least one positive example of a pedestrian are used for training and evaluation.

Training

At a resolution of 640×480 and 640×512 there will be 9,600 and 10,240 anchor boxes per image for the Caltech and KAIST datasets respectively (the anchor boxes are centred on grid points across the image at a stride of 16 pixels). The RPN of a Faster-RCNN network is typically trained by randomly selecting a balanced subset of (typically 256) anchor boxes and training only those boxes for each image. In contrast, we train using all of the anchor boxes in each image in a fully convolutional manner, similar to the semantic segmentation approaches of Long et al. [2015a], using 1 image per mini-batch. In early experiments we found that this accelerated training. The mean number of pedestrians in the KAIST dataset is around 2-3 per image. This results in a large class imbalance between foreground and background samples (anchor boxes that contain a pedestrian vs those that do not) that hampers training. We used α-balancing to counteract this; we scaled the RPN classification loss for anchor boxes depending to their ground truth class (foreground or background) by a weighting factor inversely proportional to the foreground to background ratio of the complete dataset.

RPN anchor boxes were considered to contain a pedestrian if they had an intersection-over-union (IoU) overlap of > 0.5 [Liu et al., 2016] and negative if they had an overlap of < 0.25. Anchors whose overlap was between these thresholds are considered neutral and do not contribute to the training loss. We used these thresholds in contrast to the more stringent thresholds of 0.7 and 0.3 used in most Faster R-CNN implementations as we found that relaxing them improved detection rates.

3.3.3 Inference

The detections generated by the RPN are filtered using non-maximal suppression, discarding the lowest scoring detection (classification score predicting the presence of a pedestrian) of any pair of detections whose overlap IoU is > 0.25. This is in contrast to the a threshold of 0.7 used to filter RPN proposals within a Faster R-CNN network and a threshold of 0.3 used to filter predictions from the Fast R-CNN network head.

Pre-train on Caltech dataset

Our networks were first trained using the Caltech pedestrian dataset [Dollar et al., 2009] using the protocol described above. They were trained for 10 epochs, using the Adam [Kingma and Ba, 2015] optimisation algorithm using a mini-batch size of 1 and a learning rate of 1×10^{-3} for the randomly initialised parameters belonging to the RPN network head and 1×10^{-4} for the pre-trained VGG-16 parameters.

Fine-tune on KAIST dataset

The weights of a network trained on the Caltech dataset as described above were used as a starting point for training using the KAIST dataset. Given that the RPN network head is already trained by this point, we used a learning rate of 1×10^{-4} to fine-tune all network parameters, effectively treating the entire network as pre-trained.

3.3.4 Multi-spectral fusion

We compare several methods of fusing RGB and infra-red bands within the network. For each of these experiments, a network pre-trained on the Caltech dataset (as stated above) was fine tuned using the following inputs. Our early, mid and late fusion architectures are heavily inspired by those of Jingjing Liu and Metaxas [2016].

RGB: A vanilla network uses only the RGB band as input. It was trained using the images and ground truths in the KAIST training set and evaluated on images in the test set.

$RGB * 0.5 + IR * 0.5$: The infra-red image is converted to RGB via channel replication and blended equally with the RGB image and passed to a vanilla RGB network.

Spectral edge fusion: The RGB and infra-red images are first combined using spectral edge fusion. These fused RGB images are used as input for a vanilla RGB network.

Tetra-chromatic net: The first convolutional layer from the VGG-16 backbone is modified in order to accept a 4-channel input rather than a 3-channel input. The RGB part of the convolutional kernel is copied from the original network, while the weights connected to infra-red channel are randomly initialised with a Gaussian distribution with the same mean and variance as that of the weights connected to the RGB channels. This partially pre-trained layer was trained using a learning rate of 3×10^{-4}.

Early, mid and late fusion nets: The network is bifurcated, with layers after the split remaining unchanged and layers prior to the split being duplicated in order to make two paths; one for the RGB band and one for the infra-red. The two paths are joined at the split point by concatenating the per-pixel features from each incoming branch and fused using a network-in-network [Lin et al., 2013] layer. The RGB images are passed as is as input to the RGB path while the infra-red images are converted to RGB by channel replication and passed to the infra-red path. The early, mid and late fusion nets are split after the `pool1` (first max-pooling layer after the 1st block of 3 convolutional layers), `pool4` (max-pooling layer after the 4th block) or `conv5_3` layers of the VGG-16 backbone respectively.

3.4 Evaluation

Fusion / input	Log average miss rate
RGB	19.65%
$RGB \times 0.5 + IR \times 0.5$	19.04%
Spectral edge fusion	14.76%
Tetra-chromatic network	12.57%
Early fusion network	11.70%
Mid fusion network	8.32%
Late fusion network	8.86%

Table 3.1: Table 1. Log average miss rates

Following the evaluation protocols established by Dollar et al. [2009, 2011], we summarise the effect of a variety of approaches to RGB-infra-red image fusion on our pedestrian de-

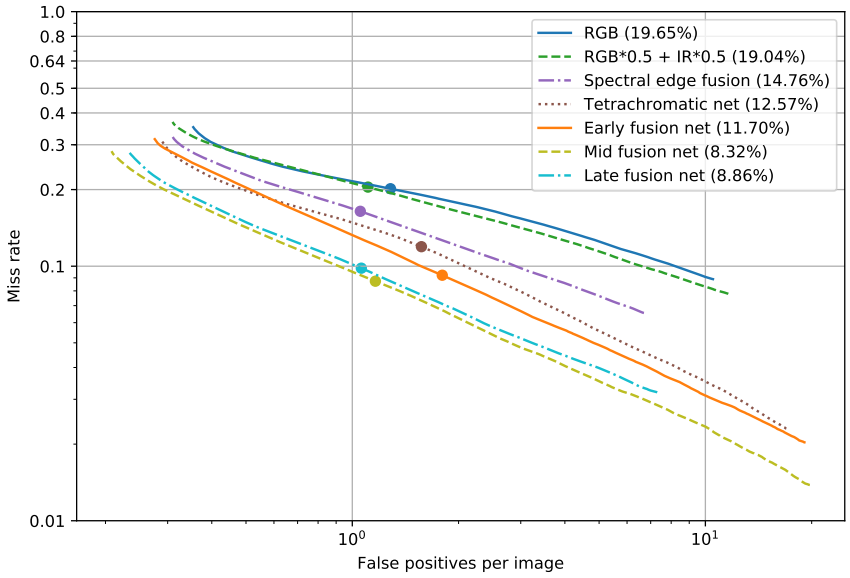


Figure 3.2: Performance evaluation of various image fusion approaches. Curves represent the trade-off between miss rate and false positive rate by varying the detection threshold. The values in parentheses in the legend are the log-average-miss-rate.

tector in Figure 3.2 and Table 3.1. Our log average miss rates are computed by averaging the miss rates corresponding to false positive rates logarithmically spaced between the values of 0.35 and 6.6 false positives per image; the range over which data is available for all approaches. This is in contrast to the range of 0.01 to 1.0 that is typically used.

The curves in Figure 3.2 fall roughly into three clusters: RGB and 50% blend; spectral edge fusion, tetra-chromatic network and early fusion; and mid and late fusion. Fusing the infra-red band into the RGB band with a 50% blend offers very little additional performance in contrast to plain RGB. Using spectral edge fusion yields a considerable improvement, reducing the miss rate from 19.04% to 14.76%. When using an un-modified network that accepts an RGB input, spectral edge fusion gives by far the best results.

Further improvements can be obtained by modifying the network structure to utilise the infra-red band as well as RGB. The miss rate is reduced to 12.57% by using a tetra-chromatic net that uses inputs with 4-channels per pixel. A slight additional improvement with a miss rate of 11.70% can be obtained by fusing after the first convolutional block. The best performance is obtained by performing fusion later in the network.

3.5 Image registration in KAIST

The KAIST [Hwang et al., 2015] dataset was captured by Hwang *et al.* using a sophisticated camera rig that employed a beam splitter to ensure that the RGB and thermal cameras

Figure 3.3: Example of the displacement between RGB and infra-red bands that can be seen in the KAIST dataset, in this case illustrated using spectral edge fusion. The horizontal disparity between the RGB image of the two pedestrians and the infra-red band can be seen in the form of the light coloured ghosting of the pedestrians offset to right of their RGB image.

observe the scene from the same point of view. In spite of this, a disparity between the RGB and infra-red bands can be observed in much of the KAIST dataset. This issue clearly manifested in the fused output from spectral edge fusion, an example of which can be seen in Figure 3.3. This suggests that image registration for multi-spectral imagery is a non-trivial problem. A potential solution would be to use a multi-spectral camera whose pixels include elements for each wavelength that is of interest. Such sensors however are uncommon and are likely to be expensive.

We hypothesize that the superior performance of mid and late fusion networks is in part due to this disparity. Feature representations generated by later layers in the network contain higher level semantic information at lower resolution. Feature image pixels will also draw information from a larger receptive field on the original image. This could simplify the process of counteracting the effects of displacement between the RGB and infra-red channels.

3.6 Discussion

In this chapter we examined the use of spectral edge fusion to fuse RGB and infra-red images as a pre-process prior to processing with a neural network. While our focus was on pedestrian detection using a network whose design is closely related to an RPN, our results are likely to carry over to other object detection settings. Spectral edge fusion improves the performance of a standard RGB pedestrian detection network when measured using the KAIST benchmark, although less so than modifying the network to use multi-spectral

imagery directly. Our exploration of object detection in this chapter will be picked up again in Chapter 7, where we use Mask R-CNN for instance segmentation. Mask R-CNN extends a Faster R-CNN object detection network with mask predictions for segmentation.

Our use of spectral edge fusion brought to our attention the displacement between the RGB and infra-red bands that can be seen throughout much of the KAIST dataset. We believe that this offers a partial explanation for the superior performance of late fusion architectures, with the higher parameter counts of late fusion architectures also contributing to our results. This suggests that better performance could be obtained using a dataset with pixel-accurate registration between RGB and thermal imagery. This also suggests spectral edge fusion as a valuable diagnostic tool for assessing the quality of registration in multi-spectral imagery.

Most importantly for practical applications beyond pedestrian detection, the misalignment between RGB and infra-red bands highlights the challenges present in constructing a multi-spectral camera rig that is to function outside laboratory conditions. In spite of the effort expended by Hwang et al. [2015] in aligning the optical components of their rig, misalignment still occurred. We hypothesize that practitioners looking to construct a similar rig for other practical applications are likely to encounter this problem.

We will return to theme of object detection in Chapter 7 when we discuss our use of object detection and instance segmentation models for detecting by-catch.

In the next chapter we focus on the area of unsupervised domain adaptation in which we attempt to train a model to achieve good performance by on unlabelled samples from a target domain, using only the supervision signal from labelled samples from a different distribution, *e.g.* artificially generated data. An effective domain adaptation algorithm could be used to train models that operate effectively on thermal imagery.

4 Consistency regularization for unsupervised visual domain adaptation

As stated in Sections 2.7 and 2.8, semi-supervised learning and unsupervised domain adaptation are related problems, with the main difference being the presence of the domain gap between the labelled source domain and unlabelled target domain samples. The potential to train using abundantly labelled synthetic data while achieving good performance on unlabelled real data is a tantalising one, hence domain adaptation is an active research area within the community. The scale of the challenge presented by a synthetic-to-real domain adaptation problem can be seen in the VisDA-17 domain adaptation challenge [Peng et al., 2018] images shown in Figure 4.1. We will present our winning solution to it in Section 4.2.2.

In this chapter we contribute a consistency regularization based domain adaptation algorithm that we presented in our paper French et al. [2018b]. The work of Tarvainen and Valpola [2017] and Sajjadi et al. [2016b] demonstrated the effectiveness of consistency regularization with random image augmentations to achieve state of the art performance in semi-supervised learning benchmarks. We modified this approach to work in a domain adaptation scenario. We will show that this can achieve excellent results in specific small image domain adaptation benchmarks. More challenging scenarios, notably MNIST $\rightarrow$ SVHN and the VisDA-17 domain adaptation challenge required further modifications. To this end, we developed confidence thresholding and class balancing loss that allowed us to achieve state of the art results in a variety of benchmarks, with some of our results coming close to those achieved by traditional supervised learning. Our approach is sufficiently flexible to be applicable to a variety of network architectures, both randomly initialized and pre-trained.

We would also like to note the concurrent work of Shu et al. [2018]. They adapt virtual adversarial training [Miyato et al., 2017] – another consistency regularization based ap-

(a) VisDa-17 training set images; the labelled source domain

(b) VisDa-17 validation set images; the unlabelled target domain

Figure 4.1: Images from the VisDA-17 domain adaptation challenge

proach – and combine it with adversarial domain adaptation [Ganin and Lempitsky, 2015], achieving small image benchmark results that are competitive with and sometimes beating our own, that we present here.

Our approach is described in Section 4.1 with our experiments and results in Section 4.2. We discuss our work in Section 4.3.

4.1 Method

Our model builds upon the mean teacher semi-supervised learning model [Tarvainen and Valpola, 2017], which we will describe. Subsequently we will present our modifications that enable domain adaptation.

The structure of the mean teacher model [Tarvainen and Valpola, 2017] – also discussed in section 2.7.4 – is shown in Figure 4.2a. The student network is trained using gradient descent, while the weights of the teacher network are an exponential moving average of those of the student. During training each input sample x_i is passed through both the student and teacher networks, generating predicted class probability vectors z_i (student) and $\tilde{z}_i$ (teacher). Different dropout, noise and image translation parameters are used for the student and teacher pathways.

During each training iteration a mini-batch of samples is drawn from the dataset, consisting of both labelled and unlabelled samples. The training loss is the sum of a supervised and an unsupervised component. The supervised loss is cross-entropy loss computed using z_i (student prediction). It is masked to 0 for unlabelled samples for which no ground truth is available. The unsupervised component is the consistency loss. It penalises the difference in class predictions between student (z_i) and teacher ($\tilde{z}_i$) networks for the same input sample. It is computed using the mean squared difference between the class probability predictions z_i and $\tilde{z}_i$.

Laine and Aila [2017] and Tarvainen and Valpola [2017] found that it was necessary to apply a time-dependent weighting to the unsupervised loss during training in order to prevent the network from getting stuck in a degenerate solution that gives poor classification performance. They used a function that follows a Gaussian curve from 0 to 1 during the first 80 epochs.

In the following subsections we will describe our contributions in detail along with the motivations for introducing them.

4.1.1 Adapting to domain adaptation

We minimise the same loss as in Tarvainen and Valpola [2017]; we apply cross-entropy loss to labelled source samples and unsupervised consistency loss to target samples. As in Tarvainen and Valpola [2017], consistency loss is computed as the mean-squared difference between predictions produced by the student (z_{Ti}) and teacher ($\tilde{z}_{Ti}$) networks with different augmentation, dropout and noise parameters.

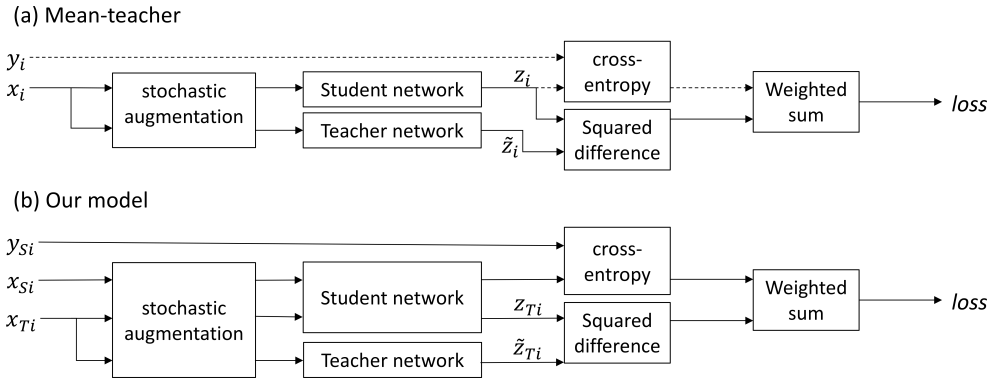


Figure 4.2: The network structures of the original mean teacher model and our model. Dashed lines in the mean teacher model indicate that ground truth labels – and therefore cross-entropy classification loss – are only available for labelled samples.

The models of Tarvainen and Valpola [2017] and of Laine and Aila [2017] were designed for semi-supervised learning problems in which a subset of the samples in a single dataset have ground truth labels. During training both models mix labelled and unlabelled samples together in a mini-batch. In contrast, unsupervised domain adaptation problems use two distinct datasets with different underlying distributions; labelled source and unlabelled target. Our variant of the mean teacher model – shown in Figure 4.2b – has separate source (X_{Si}) and target (X_{Ti}) paths. Inspired by Li et al. [2016], we process mini-batches from the source and target datasets separately (per iteration) so that batch normalization uses different normalization statistics for each domain during training.[1]. We do not use the approach of Li et al. [2016] as-is, as they handle the source and target datasets separately in two distinct training phases, where our approach must train using both simultaneously. We also do not maintain separate exponential moving averages of the means and variances for each dataset for use at test time.

As seen in the 'MT+TF' row of Table 4.5, the model described thus far achieves state of the art results in 5 out of 8 small image benchmarks. The MNIST → SVHN, STL → CIFAR-10 and Syn-digits → SVHN benchmarks however require additional modifications to achieve good performance.

4.1.2 Confidence thresholding

We found that replacing the Gaussian ramp-up factor that scales the unsupervised loss with confidence thresholding stabilized training in more challenging domain adaptation scenarios. For each unlabelled sample x_{Ti} the teacher network produces the predicted class probability vector $\tilde{z}_{Tij}$ – where j is the class index drawn from the set of classes C – from which we compute the confidence $\tilde{f}_{Ti} = \max_{j \in C}(\tilde{z}_{Tij})$; the predicted probability of the predicted class of the sample. If $\tilde{f}_{Ti}$ is below the confidence threshold (a parameter search found 0.968 to

[1]This is simple to implement using most neural network tool-kits; evaluate the network once for source samples and a second time for target samples, compute the supervised and unsupervised losses respectively and combine.

be an effective value for small image benchmarks), the consistency loss for the sample x_i is masked to 0.

Our working hypothesis is that confidence thresholding acts as a filter, shifting the balance in favour of the student learning correct labels from the teacher. While high network prediction confidence does not guarantee correctness there is a positive correlation. Given the tolerance to incorrect labels reported in Laine and Aila [2017], we believe that the higher signal-to-noise ratio underlies the success of this component of our approach.

The use of confidence thresholding achieves a state of the art results in the STL $\rightarrow$ CIFAR-10 and Syn-digits $\rightarrow$ SVHN benchmarks, as seen in the 'MT+CT+TF' row of Table 4.5. While confidence thresholding can result in very slight reductions in performance (see the MNIST $\leftrightarrow$ USPS and SVHN $\rightarrow$ MNIST results), its ability to stabilise training in challenging scenarios leads us to recommend it as a replacement for the time-dependent Gaussian ramp-up used in Laine and Aila [2017].

4.1.3 Data augmentation

We explored the effect of three data augmentation schemes in our small image benchmarks (section 4.2.1). Our minimal scheme (that should be applicable in non-visual domains) consists of Gaussian noise (with $\sigma = 0.1$) added to the pixel values. The standard scheme (indicated by 'TF' in Table 4.5) was used in Laine and Aila [2017] and adds translations in the interval $[-2, 2]$ and horizontal flips for the CIFAR-10 $\leftrightarrow$ STL experiments. The affine scheme (indicated by 'TFA') adds random affine transformations defined by the matrix in (4.1), where $\mathcal{N}(0, 0.1)$ denotes a real value drawn from a normal distribution with mean 0 and standard deviation 0.1.

$$\begin{bmatrix} 1 + \mathcal{N}(0, 0.1) & \mathcal{N}(0, 0.1) \\ \mathcal{N}(0, 0.1) & 1 + \mathcal{N}(0, 0.1) \end{bmatrix} \tag{4.1}$$

The use of translations and horizontal flips has a significant impact in a number of our benchmarks. It is necessary in order to outpace prior art in the MNIST $\leftrightarrow$ USPS and SVHN $\rightarrow$ MNIST benchmarks and improves performance in the CIFAR-10 $\leftrightarrow$ STL benchmarks. The use of affine augmentation can improve performance in experiments involving digit and traffic sign recognition datasets, as seen in the 'MT+CT+TFA' row of Table 4.5. In contrast it can impair performance when used with photographic datasets, as seen in the the STL $\rightarrow$ CIFAR-10 experiment. It also impaired performance in the VisDA-17 experiment (section 4.2.2).

4.1.4 Class balance loss

With the adaptations made so far the challenging MNIST $\rightarrow$ SVHN benchmark remains undefeated due to training instabilities. During training we noticed that the error rate on the SVHN test set decreases at first, then rises and reaches high values before training completes. We diagnosed the problem by recording the predictions for the SVHN target domain samples after each epoch. The rise in error rate correlated with the predictions

evolving toward a condition in which most samples are predicted as belonging to the '1' class; the most populous class in the SVHN dataset. We hypothesize that the class imbalance in the SVHN dataset caused the unsupervised loss to reinforce the '1' class more often than the others, resulting in the network settling in a degenerate local minimum. Rather than distinguish between digit classes as intended it separated MNIST from SVHN samples and assigned the latter to the '1' class.

We addressed this problem by introducing a class balance loss term that penalises the network for making predictions that exhibit large class imbalance. For each target domain mini-batch we compute the mean of the predicted sample class probabilities over the sample dimension, resulting in the mini-batch mean per-class probability. The loss is computed as the binary cross entropy between the mean class probability vector and a uniform probability vector. We balance the strength of the class balance loss with that of the consistency loss by multiplying the class balance loss by the average of the confidence threshold mask (e.g. if 75% of samples in a mini-batch pass the confidence threshold, then the class balance loss is multiplied by 0.75).[2]

We would like to note the similarity between our class balance loss and the entropy maximisation loss in the IMSAT clustering model of Hu et al. [2017]; IMSAT employs entropy maximisation to encourage uniform cluster sizes and entropy minimisation to encourage unambiguous cluster assignments.

4.2 Experiments

We perform two sets of experiments. Firstly we apply our approach to small image datasets using simple convolutional network architectures. Following this we apply residual networks [He et al., 2016] to the more challenging VisDA 2017 [Peng et al., 2018] challenge dataset.

Our implementation was developed using PyTorch [Chintala et al., 2017] and is publicly available; our small image experiments can be found at `http://github.com/Britefury/self-ensemble-visual-domain-adapt` and our VisDA 2017 experiments can be found at `http://github.com/Britefury/self-ensemble-visual-domain-adapt-photo`.

4.2.1 Small image datasets

The small image datasets used in these experiments are described in Table 4.1.

Data preparation

Some of the experiments that involved datasets described in Table 4.1 required additional data preparation in order to match the resolution and format of the input samples and match the classification target. These additional steps will now be described.

[2] We expect that class balance loss is likely to adversely affect performance on target datasets with large class imbalance.

	# train	# test	# classes	Target	Resolution	Channels
USPS[a]	7,291	2,007	10	Digits	16×16	Mono
MNIST	60,000	10,000	10	Digits	28×28	Mono
SVHN	73,257	26,032	10	Digits	32×32	RGB
CIFAR-10	50,000	10,000	10	Object ID	32×32	RGB
STL[b]	5,000	8,000	10	Object ID	96×96	RGB
Syn-Digits[c]	479,400	9,553	10	Digits	32×32	RGB
Syn-Signs	100,000	–	43	Traffic signs	40×40	RGB
GTSRB	32,209	12,630	43	Traffic signs	*varies*	RGB

[a] Available from `http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/zip.train.gz` and `http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/zip.test.gz`
[b] Available from `http://ai.stanford.edu/~acoates/stl10/`
[c] Available from Ganins' website at `http://yaroslav.ganin.net/`

Table 4.1: Small image dataset summary

MNIST ↔ USPS The USPS images were up-scaled using bilinear interpolation from 16×16 to 28×28 resolution to match that of MNIST.

CIFAR-10 ↔ STL CIFAR-10 and STL are both 10-class image datasets. The STL images were down-scaled to 32×32 resolution to match that of CIFAR-10. The 'frog' class in CIFAR-10 and the 'monkey' class in STL were removed as they have no equivalent in the other dataset, resulting in a 9-class problem with 10% less samples in each dataset.

Syn-Signs → GTSRB GTSRB is composed of images that vary in size and come with annotations that provide region of interest (bounding box around the sign) and ground truth classification. We extracted the region of interest from each image and scaled them to a resolution of 40×40 to match those of Syn-Signs.

MNIST ↔ SVHN The MNIST images were padded to 32×32 resolution and converted to RGB by replicating the grey-scale channel into the three RGB channels to match the format of SVHN.

Network architectures

Our small image network architectures are shown in Tables 4.2 - 4.4.

Description	Shape
28×28 Mono image	$28 \times 28 \times 1$
Conv. $5 \times 5 \times 32$, batch norm	$24 \times 24 \times 32$
Max-pool, 2x2	$12 \times 12 \times 32$
Conv. $3 \times 3 \times 64$, batch norm	$10 \times 10 \times 64$
Conv. $3 \times 3 \times 64$, batch norm	$8 \times 8 \times 64$
Max-pool, 2x2	$4 \times 4 \times 64$
Dropout, 50%	$4 \times 4 \times 64$
Fully connected, 256 units	256
Fully connected, 10 units, softmax	10

Table 4.2: MNIST ↔ USPS architecture

Description	Shape
32×32 RGB image	$32 \times 32 \times 3$
Conv. $3 \times 3 \times 128$, pad 1, batch norm	$32 \times 32 \times 128$
Conv. $3 \times 3 \times 128$, pad 1, batch norm	$32 \times 32 \times 128$
Conv. $3 \times 3 \times 128$, pad 1, batch norm	$32 \times 32 \times 128$
Max-pool, 2x2	$16 \times 16 \times 128$
Dropout, 50%	$16 \times 16 \times 128$
Conv. $3 \times 3 \times 256$, pad 1, batch norm	$16 \times 16 \times 256$
Conv. $3 \times 3 \times 256$, pad 1, batch norm	$16 \times 16 \times 256$
Conv. $3 \times 3 \times 256$, pad 1, batch norm	$16 \times 16 \times 256$
Max-pool, 2x2	$8 \times 8 \times 256$
Dropout, 50%	$8 \times 8 \times 256$
Conv. $3 \times 3 \times 512$, pad 0, batch norm	$6 \times 6 \times 512$
Conv. $1 \times 1 \times 256$, batch norm	$6 \times 6 \times 256$
Conv. $1 \times 1 \times 128$, batch norm	$6 \times 6 \times 128$
Global pooling layer	$1 \times 1 \times 128$
Fully connected, 10 units, softmax	10

Table 4.3: MNIST $\leftrightarrow$ SVHN, CIFAR-10 $\leftrightarrow$ STL and Syn-Digits $\rightarrow$ SVHN architecture

Description	Shape
40×40 RGB image	$40 \times 40 \times 3$
Conv. $3 \times 3 \times 96$, pad 1, batch norm	$40 \times 40 \times 96$
Conv. $3 \times 3 \times 96$, pad 1, batch norm	$40 \times 40 \times 96$
Conv. $3 \times 3 \times 96$, pad 1, batch norm	$40 \times 40 \times 96$
Max-pool, 2x2	$20 \times 20 \times 96$
Dropout, 50%	$20 \times 20 \times 96$
Conv. $3 \times 3 \times 192$, pad 1, batch norm	$20 \times 20 \times 192$
Conv. $3 \times 3 \times 192$, pad 1, batch norm	$20 \times 20 \times 192$
Conv. $3 \times 3 \times 192$, pad 1, batch norm	$20 \times 20 \times 192$
Max-pool, 2x2	$10 \times 10 \times 192$
Dropout, 50%	$10 \times 10 \times 192$
Conv. $3 \times 3 \times 384$, pad 1, batch norm	$10 \times 10 \times 384$
Conv. $3 \times 3 \times 384$, pad 1, batch norm	$10 \times 10 \times 384$
Conv. $3 \times 3 \times 384$, pad 1, batch norm	$10 \times 10 \times 384$
Max-pool, 2x2	$5 \times 5 \times 384$
Dropout, 50%	$5 \times 5 \times 384$
Global pooling layer	$1 \times 1 \times 384$
Fully connected, 43 units, softmax	43

Table 4.4: Syn-signs $\rightarrow$ GTSRB architecture

Training procedure

Our networks were trained for 300 epochs. We used the Adam [Kingma and Ba, 2015] gradient descent algorithm with a learning rate of 0.001. We trained using mini-batches composed of 256 samples, except in the Syn-digits $\rightarrow$ SVHN and Syn-signs $\rightarrow$ GTSRB experiments where we used 128 in order to reduce memory usage. The consistency loss was weighted by a factor of 3 and the class balancing loss was weighted by 0.005. Our teacher network weights t_i were updated so as to be an exponential moving average of those of the student s_i using the formula $t_i = \alpha t_{i-1} + (1 - \alpha)s_i$, with a value of 0.99 for α. A complete pass over the target dataset was considered to be one epoch in all experiments except the

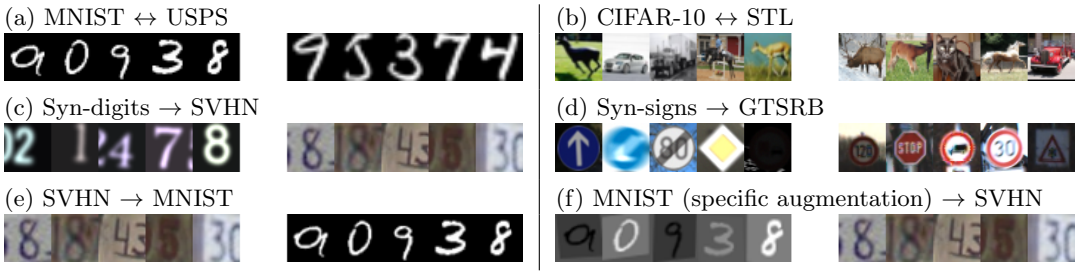


Figure 4.3: Small image domain adaptation example images

MNIST → USPS and CIFAR-10 → STL experiments due to the small size of the target datasets, in which case one epoch was considered to be a pass over the larger source dataset. We note that only the training sets of the small image datasets were used during training; the test sets used for reporting scores only.

We found that using the proportion of samples that passed the confidence threshold can be used to drive early stopping [Prechelt, 1998]. The final score was the target test set performance at the epoch at which the highest confidence threshold pass rate was obtained.

Results

Our results can be seen in Table 4.5. The 'train on source' and 'train on target' results report the target domain performance of supervised training on the source and target domains. They represent the expected baseline and best achievable result. The 'Specific aug.' experiments used data augmentation specific to the MNIST → SVHN adaptation path that is discussed further down. The same network architectures and augmentation parameters were used for domain adaptation experiments and the supervised baselines.

MNIST ↔ USPS (see Figure 4.3a). MNIST and USPS are both grey-scale hand-written digit datasets. In both adaptation directions our approach not only demonstrates a significant improvement over prior art but nearly achieves the performance of supervised learning using the target domain ground truths. The strong performance of the base mean teacher model can be attributed to the similarity of the datasets to one another. It is worth noting that data augmentation allows our 'train on source' baseline to outpace prior domain adaptation methods.

CIFAR-10 ↔ STL (see Figure 4.3b). CIFAR-10 and STL are both 10-class image datasets, though as stated earlier we removed the frog and monkey class from each respectively. We obtained strong performance in the STL → CIFAR-10 path, but only by using confidence thresholding. The CIFAR-10 → STL results are more interesting; the 'train on source' baseline performance outperforms that of a network trained on the STL target domain, most likely due to the small size of the STL training set. Our domain adaptation results outpace both the baseline performance and the 'theoretical maximum' of a network trained on the target domain, lending further evidence to the view of Sajjadi et al. [2016b] and Laine and Aila [2017] that consistency regularization acts as an effective unsupervised regularizer.

Syn-Digits → SVHN (see Figure 4.3c). The Syn-Digits dataset is a synthetic dataset designed by Ganin and Lempitsky [2015] to be used as a source dataset in domain adaptation experiments with SVHN as the target dataset. Other approaches have achieved good scores on this benchmark, beating the baseline by a significant margin. Our result improves on them, reducing the error rate from 6.9% to 2.9%; even slightly outpacing the 'train on target' 3.4% error rate achieved using supervised learning.

Syn-Signs → GTSRB (see Figure 4.3d). Syn-Signs is another synthetic dataset designed by Ganin and Lempitsky [2015] to target the 43-class GTSRB Stallkamp et al. [2011] (German Traffic Signs Recognition Benchmark) dataset. Our approach halved the best error rate of competing approaches. Once again, our approaches slightly outpaces the 'train on target' supervised learning upper bound.

SVHN → MNIST (see Figure 4.3e). Google's SVHN (Street View House Numbers) is a colour digits dataset of house number plates. Our approach significantly out-paces other techniques and achieves an accuracy close to that of supervised learning.

MNIST → SVHN (see Figure 4.3f). This adaptation path is somewhat more challenging as MNIST digits are greyscale and uniform in terms of size, aspect ratio and intensity range, in contrast to the variably sized colour digits present in SVHN. As a consequence, adapting from MNIST to SVHN required additional work. Class balancing loss was necessary to ensure training stability and additional experiment specific data augmentation was required to achieve good accuracy. The use of translations and affine augmentation (see section 4.1.3) results in an accuracy score of 37%. Significant improvements resulted from additional augmentation in the form of random intensity flips (negative image), and random intensity scales and offsets drawn from the intervals $[0.25, 1.5]$ and $[-0.5, 0.5]$ respectively. These hyper-parameters were selected in order to augment MNIST samples to match the intensity variations present in SVHN, as illustrated in Figure 4.3f. With these additional modifications, we achieve a result that significantly outperforms prior art and nearly achieves the accuracy of a supervised classifier trained on the target dataset. We found that applying these additional augmentations to the source MNIST dataset only yielded good results; applying them to the target SVHN dataset as well yielded a small improvement but was not essential. It should also be noted that this augmentation scheme raises the performance of the 'train on source' baseline to just above that of much of the prior art.

4.2.2 VisDA-2017 visual domain adaptation challenge

The VisDA-2017 image classification challenge [Peng et al., 2018] is a 12-class domain adaptation problem consisting of three datasets: a training set consisting of 3D renderings of Sketchup models, and validation and test sets consisting of real images (see Figure 4.1) drawn from the COCO [Lin et al., 2014] and YouTube BoundingBoxes [Real et al., 2017] datasets respectively. The objective is to learn from labelled computer generated images and correctly predict the class of real images. Ground truth labels were made available for the training and validation sets only; test set scores were computed by a server operated by the competition organisers.

Training procedure

While the algorithm is that presented above, we base our network on the pre-trained ResNet-152 [He et al., 2016] network provided by PyTorch [Chintala et al., 2017], rather than using a randomly initialised network as before. The final 1000-class classification layer is removed and replaced with two fully-connected layers; the first has 512 units with a ReLU non-linearity while the final layer has 12 units with a softmax non-linearity. Results from our original competition submissions and newer results using two data augmentation schemes are presented in Table 4.6. Our reduced augmentation scheme consists of random crops, random horizontal flips and random uniform scaling. It is very similar to scheme used for ImageNet image classification in He et al. [2016]. Our competition configuration includes additional augmentation that was specifically designed for the VisDA dataset, although we subsequently found that it makes little difference. Our augmentation schemes are detailed below.

Reduced data augmentation:

- scale image so that its smallest dimension is 176 pixels, then randomly crop a 160×160 section from the scaled image
- *No* random affine transformations as they increase confusion between the car and truck classes in the validation set
- random uniform scaling in the range $[0.75, 1.333]$
- horizontal flipping

Competition data augmentation adds the following in addition to the above:

- random intensity/brightness scaling in the range $[0.75, 1.333]$
- random rotations, normally distributed with a standard deviation of 0.2π
- random desaturation in which the colours in an image are randomly desaturated to grey-scale by a factor between 0% and 100%
- rotations in colour space, around a randomly chosen axes with a standard deviation of 0.05π
- random offset in colour space, after standardisation using parameters specified by PyTorch implementation of ResNet-152

Our training procedure was the same as that used in the small image experiments, except that we used 160×160 images, a batch size of 56 (reduced from 64 to fit within the memory of an nVidia 1080-Ti), a consistency loss weight of 10 (instead of 3), a confidence threshold of 0.9 (instead of 0.968) and a class balancing weight of 0.01. We used the Adam [Kingma and Ba, 2015] gradient descent algorithm with a learning rate of 10^{-5} for the final two randomly initialized layers and 10^{-6} for the pre-trained layers. The first convolutional layer and the first group of convolutional layers (with 64 feature channels) of the pre-trained ResNet were left unmodified during training.

Results

It is worth noting that we applied test time augmentation (we averaged predictions form 16 differently augmented images) to achieve our competition results. We present results with and without test time augmentation in Table 4.6. Our VisDA competition test set score is also the result of ensembling the predictions of 5 different networks. Full per-class results are presented in Tables 4.7 and 4.8.

4.3 Discussion

In this chapter we have presented an effective domain adaptation algorithm that has achieved state of the art results in a number of benchmarks and has achieved accuracies that are almost on par with traditional supervised learning on digit recognition benchmarks targeting the MNIST and SVHN datasets. The resulting networks will exhibit strong performance on samples from both the source and target domains. Our approach is sufficiently flexible to be usable for a variety of network architectures, including those based on randomly initialised and pre-trained networks. We have thus demonstrated that consistency regularization is a powerful and flexible method that can work for domain adaptation in addition to semi-supervised learning.

Miyato et al. [2017] stated that the consistency regularization methods of Laine and Aila [2017] – on which our algorithm is based – operate by label propagation. This view is supported by our results, in particular our MNIST → SVHN experiment. The latter requires additional intensity augmentation in order to sufficiently align the dataset distributions, after which good quality label predictions are propagated throughout the target dataset. In cases where data augmentation is insufficient to align the dataset distributions, a pre-trained network may be used to bridge the gap, as in our solution to the VisDA-17 challenge. This leads us to conclude that effective domain adaptation can be achieved by first aligning the distributions of the source and target datasets – the focus of much prior art in the field – and then refining their correspondence; a task to which consistency regularization is well suited.

The work presented in this chapter gave us valuable experience in applying consistency regularization. While the focus was on domain adaptation, our exploration of driving it with data augmentation and the use of confidence thresholding proved to be very useful for subsequent work.

The next chapter covers the application of consistency regularization to the problem of semi-supervised semantic segmentation. The use of consistency regularization for semi-supervised classification in prior work and its successful application in the work presented in this chapter led us to consider it as a likely candidate approach for the semantic segmentation problem. The Mean Teacher algorithm [Tarvainen and Valpola, 2017] and the confidence thresholding technique that we presented in Section 4.1.2 proved to be key the success of our semi-supervised semantic segmentation approach and also contributed to the work discussed in Chapter 6.)

In Chapter 7 we find that the fisheries surveillance footage that our by-catch quantifier must analyse with exhibits a variety of capture conditions with differing visual appearances. This suggests that an effective domain adaptation algorithm could prove to be very useful.

	USPS	MNIST	SVHN	MNIST	CIFAR	STL	Syn Digits	Syn Signs
	–	–	–	–	–	–	–	–
	MNIST	USPS	MNIST	SVHN	STL	CIFAR	SVHN	GTSRB
TRAIN ON SOURCE								
SupSrc*	77.55 ±0.36	82.03 ±0.52	66.5 ±0.86	25.44 ±1.25	72.84 ±0.27	51.88 ±0.64	86.86 ±0.38	96.95 ±0.16
SupSrc+TF	77.53 ±2.07	95.39 ±0.42	68.65 ±0.67	24.86 ±1.47	75.2 ±0.13	59.06 ±0.46	87.45 ±0.29	97.3 ±0.07
SupSrc+TFA	91.97 ±0.96	96.25 ±0.24	71.73 ±2.56	28.69 ±0.71	75.18 ±0.34	59.38 ±0.26	87.16 ±0.38	98.02 ±0.09
Specific aug.[b]	–	–	–	61.99 ±1.74	–	–	–	–
RevGrad[a] [1]	74.01	91.11	73.91	35.67	66.12	56.91	91.09	88.65
DCRN [2]	73.67	91.8	81.97	40.05	66.37	58.65	–	–
G2A [3]	90.8	92.5	84.70	36.4	–	–	–	–
ADDA [4]	90.1	89.4	76.00	–	–	–	–	–
ATT [5]	–	–	86.20	52.8	–	–	93.1	96.2
SBADA-GAN [6]	97.60	95.04	76.14	61.08	–	–	–	–
ADA [7]	–	–	97.6	–	–	–	91.86	97.66
OUR RESULTS								
MT+TF	98.07 ±1.26	**98.26** ±0.05	99.18 ±0.05	13.96[c] ±1.97	80.08 ±0.11	18.3 ±4.04	15.94 ±0.00	98.63 ±0.04
MT+CT*	92.35 ±3.85	88.14 ±0.15	93.33 ±2.63	33.87[c] ±1.80	77.53 ±0.05	71.65 ±0.30	96.01 ±0.04	98.53 ±0.07
MT+CT+TF	97.28 ±1.23	98.13 ±0.08	98.64 ±0.19	34.15[c] ±1.59	79.73 ±0.20	**74.24** ±0.21	96.51 ±0.04	98.66 ±0.05
MT+CT+TFA	**99.54** ±0.02	98.23 ±0.06	**99.26** ±0.02	37.49[c] ±1.09	**80.09** ±0.14	69.86 ±0.88	**97.11** ±0.02	**99.37** ±0.04
Specific aug.[b]	–	–	–	**97.0**[c] ±0.03	–	–	–	–
TRAIN ON TARGET								
SupTgt*	99.53 ±0.01	97.29 ±0.09	99.59 ±0.04	95.7 ±0.06	67.75 ±1.00	88.86 ±0.17	95.62 ±0.09	98.49 ±0.14
SupTgt+TF	99.62 ±0.02	97.65 ±0.08	99.61 ±0.02	96.19 ±0.04	70.98 ±0.35	89.83 ±0.17	96.18 ±0.04	98.64 ±0.04
SupTgt+TFA	99.62 ±0.01	97.83 ±0.08	99.59 ±0.03	96.65 ±0.05	70.03 ±0.51	90.44 ±0.17	96.59 ±0.04	99.22 ±0.10
Specific aug.[b]	–	–	–	97.16 ±0.02	–	–	–	–

[1] Ganin and Lempitsky [2015], [2] Ghifary et al. [2016], [3] Sankaranarayanan et al. [2017], [4] Tzeng et al. [2017], [5] Saito et al. [2017a], [6] Russo et al. [2018], [7] Haeusser et al. [2017]

[a] RevGrad results were available in both Ganin and Lempitsky [2015] and Ghifary et al. [2016]; we drew results from both papers to obtain results for all of the experiments shown.

[b] MNIST → SVHN specific intensity augmentation as described in Section 4.2.1.

[c] MNIST → SVHN experiments used class balance loss.

Table 4.5: Small image benchmark classification accuracy; each result is presented as *mean* ± *standard error*, computed from 5 independent runs. The abbreviations for components of our models are as follows: MT = mean teacher, CT = confidence thresholding, TF = translation and horizontal flip augmentation, TFA = translation, horizontal flip and affine augmentation, * indicates minimal augmentation.

VALIDATION PHASE		TEST PHASE	
Team / model	Mean class acc.	Team / model	Mean class acc.
OTHER TEAMS			
bchidlovski [1]	**83.1**	NLE-DA [1]	87.7
BUPT_OVERFIT	77.8	BUPT_OVERFIT	85.4
Uni. Tokyo MIL [2]	75.4	Uni. Tokyo MIL [2]	82.4
OUR COMPETITION RESULTS			
ResNet-50 model	82.8[a]	ResNet-152 model	**92.8**[ab]
OUR NEWER RESULTS (all using ResNet-152)			
Minimal aug.[*]	74.2 ±0.38	Minimal aug.[*]	77.52 ±0.78
Reduced aug.	85.4 ±0.09	Reduced aug.	91.17 ±0.08
+ test time aug.	**86.6** ±0.08[a]	+ test time aug.	92.25 ±0.09[a]
Competition config.	84.29 ±0.11	Competition config.	91.14 ±0.06
+ test time aug.	85.52 ±0.13[a]	+ test time aug.	92.41 ±0.07[a]

[1] Csurka et al. [2017], [2] Saito et al. [2017b]
[a] Used test-time augmentation; averaged predictions of 16 differently augmentations versions of each image
[b] Our competition submission ensembled predictions from 5 independently trained networks

Table 4.6: VisDA-17 performance, presented as *mean* ± *std-err* of 5 independent runs.

	Plane	Bicycle	Bus	Car	Horse	Knife	
COMPETITION RESULTS							
ResNet-50	96.3	87.9	84.7	55.7	95.9	95.2	
NEWER RESULTS (ResNet-152)							
Minimal aug	92.94 ±0.23	84.88 ±0.33	71.56 ±1.38	41.24 ±0.45	88.85 ±0.59	92.40 ±0.51	
Reduced aug	96.19 ±0.08	87.83 ±0.72	84.38 ±0.41	66.47 ±2.03	96.07 ±0.13	96.06 ±0.28	
+ test time aug	97.13 ±0.08	89.28 ±0.65	84.93 ±0.49	67.67 ±2.08	96.54 ±0.16	97.48 ±0.19	
Competition config.	95.93 ±0.13	87.36 ±0.53	85.22 ±0.38	58.56 ±0.81	96.23 ±0.08	95.65 ±0.27	
+ test time aug	96.89 ±0.14	89.06 ±0.55	85.51 ±0.37	59.73 ±0.88	96.59 ±0.06	97.55 ±0.21	
	M.cycle	Person	Plant	Sk.brd	Train	Truck	**Mean Class Acc.**
COMPETITION RESULTS							
ResNet-50	88.6	77.4	93.3	92.8	87.5	38.2	82.8
NEWER RESULTS (ResNet-152)							
Minimal aug	67.51 ±0.80	63.46 ±0.77	84.47 ±0.55	71.84 ±2.41	83.22 ±0.33	48.09 ±0.63	74.20 ±0.38
Reduced aug	90.49 ±0.12	81.45 ±0.40	95.27 ±0.16	91.48 ±0.34	87.54 ±0.52	51.60 ±1.05	85.40 ±0.09
+ test time aug	90.99 ±0.17	83.33 ±0.41	96.12 ±0.14	94.69 ±0.32	88.53 ±0.54	52.54 ±1.26	86.60 ±0.11
Competition config.	90.60 ±0.48	80.03 ±0.55	94.79 ±0.16	90.77 ±0.29	88.42 ±0.39	47.90 ±0.97	84.29 ±0.11
+ test time aug	91.00 ±0.52	81.59 ±0.54	95.58 ±0.17	94.29 ±0.28	89.28 ±0.38	49.21 ±1.01	85.52 ±0.13

Table 4.7: Full VisDA-17 validation set results

	Plane	Bicycle	Bus	Car	Horse	Knife	
COMPETITION RESULTS (ensemble of 5 models)							
ResNet-152	96.9	92.4	92.0	97.2	95.2	98.8	
NEWER RESULTS (ResNet-152)							
Minimal aug	88.44	84.80	75.08	84.08	79.95	72.62	
	±0.61	±0.81	±0.73	±1.02	±0.86	±3.57	
Reduced aug	95.63	89.90	91.44	96.18	94.17	96.51	
	±0.27	±0.29	±0.15	±0.28	±0.11	±0.18	
+ test time aug	96.72	91.67	92.21	96.41	94.72	98.03	
	±0.26	±0.33	±0.20	±0.29	±0.09	±0.18	
Competition config.	95.13	90.09	91.21	96.94	94.39	96.87	
	±0.17	±0.17	±0.37	±0.15	±0.21	±0.15	
+ test time aug	96.48	91.96	91.92	97.22	95.12	98.44	
	±0.14	±0.17	±0.29	±0.16	±0.23	±0.06	

	M.cycle	Person	Plant	Sk.brd	Train	Truck	**Mean Class Acc.**
COMPETITION RESULTS (ensemble of 5 models)							
ResNet-152	86.3	75.3	97.7	93.3	94.5	93.3	92.8
NEWER RESULTS (ResNet-152)							
Minimal aug	63.60	56.59	95.40	73.79	77.57	78.33	77.52
	±0.69	±0.77	±0.23	±2.43	±0.79	±1.40	±0.35
Reduced aug	85.02	71.31	97.35	91.11	92.42	93.03	91.17
	±0.37	±0.43	±0.22	±0.47	±0.21	±0.16	±0.08
+ test time aug	85.40	73.19	97.84	93.53	93.31	93.91	92.25
	±0.48	±0.38	±0.20	±0.32	±0.16	±0.17	±0.09
Competition config.	85.12	70.78	97.22	90.39	93.18	92.38	91.14
	±0.58	±0.68	±0.08	±0.29	±0.22	±0.23	±0.06
+ test time aug	85.75	74.06	97.77	92.91	94.21	93.09	92.41
	±0.54	±0.76	±0.07	±0.20	±0.23	±0.20	±0.07

Table 4.8: Full VisDA-17 test set results

5 Why semi-supervised semantic segmentation is challenging and how to crack it with CutMix

Viewing the task of semantic segmentation as pixel classification suggests that semi-supervised classification approaches could be adapted for segmentation. Given the simplicity of consistency regularization, its success in semi-supervised learning (see Section 2.7.4) and our successful application to the problem of unsupervised domain adaptation in Chapter 4, consistency regularization seem like a logical approach.

Our early attempts however were unsuccessful. We also note that the only reports of consistency regularization being successfully applied to segmentation are for specific problem domains. There are a couple from the medical imaging community, namely Perone and Cohen-Adad [2018] and Li et al. [2018b]. For natural photographic images only one positive report exists; that of Ji et al. [2019] who develop a semi-supervised over-clustering approach for natural photographic images, where the list of ground truth classes is highly constrained. This motivated us to analyse the problem in order to understand the reasons underlying these successes and failures, hopefully illuminating a way forward.

A significant body of prior work [Ouali et al., 2020; Luo et al., 2018; Sajjadi et al., 2016a; Shu et al., 2018; Verma et al., 2019] states that the cluster assumption [Chapelle and Zien, 2005] is key to the successful application of semi-supervised learning. In our paper [French et al., 2020a] on which we base this chapter we analysed the data distribution of semantic segmentation problems and found that the cluster assumption does not apply, suggesting that semi-supervised semantic segmentation is a particularly challenging problem.

While we believe our original analysis to be sound, we now present an alternative explanation confirmed by further results. Motivated by the recent self-supervised learning results of Chen et al. [2020a], we hypothesize that the network can minimize the consistency loss term employed in our semi-supervised approach by focusing on colour statistics rather than image content. We conduct further experiments that demonstrate that the colour augmentation scheme commonly used in self-supervised learning [Chen et al., 2020a; He et al., 2020] offers an effective solution, verifying our new hypothesis. This was presented in our later work French. and Mackiewicz. [2022]. This indicates that the data distribution does not in fact impact the performance of consistency regularization in semantic segmentation problems and that the cluster assumption is *not* necessary for effective semi-supervised learning. This is surprising given the extensive work in the literature stating its importance.

This chapter follows the work in our two papers; we make three contributions; an analysis of the data distribution of semantic segmentation, an exploration of the use of colour aug-

mentations and a semi-supervised learning approach that yields state of the art results (as of August 2020).

Our implementation is available at `https://github.com/Britefury/cutmix-semisup-seg`.

5.1 Overview

The effectiveness of consistency regularization is often attributed to the *smoothness assumption* [Luo et al., 2018] or *cluster assumption* [Chapelle and Zien, 2005; Ouali et al., 2020; Sajjadi et al., 2016a; Shu et al., 2018; Verma et al., 2019]. The smoothness assumption states that samples close to each other are likely to have the same label. The cluster assumption — a special case of the smoothness assumption — states that decision surfaces should lie in low density regions of the data distribution. This typically holds in classification tasks, where most successes of consistency regularization have been reported so far.

At a high level, semantic segmentation is classification, where each pixel is classified given its neighbourhood. In Section 5.3 we make the observation that the L^2 pixel content distance between patches centred on neighbouring pixels varies smoothly even when the class of the centre pixel changes, and thus there are no low-density regions along class boundaries. This alarming observation leads us to investigate the conditions that can allow consistency regularization to operate in these circumstances.

We start by developing a simple toy 2D classification problem in Section 5.3.1 that we first use to confirm the benefit of the clustered data. Subsequently we find that appropriately constrained consistency regularization can operate on data with a continuous distribution, where no low density regions are available to guide consistency regularization. This leads us to propose requirements for using consistency regularization for semantic segmentation problems. We find mask-based augmentation strategies to be effective, with an adapted variant of CutMix [Yun et al., 2019] realizing significant gains in Section 5.4. Furthermore, in Section 5.5 we analyze the use of colour augmentation and find that it improves performance of CutMix and allows other semi-supervised regularizers to function in semantic segmentation problems, where without it they would not.

5.2 The benefit of the cluster assumption

Consistency regularization adds a consistency loss term L_{cons} to the loss that is minimized during training [Oliver et al., 2018]. In a classification task, L_{cons} measures a distance $d(\cdot, \cdot)$ between the predictions resulting from applying a neural network f_θ to an unsupervised sample x and a perturbed version $\hat{x}$ of the same sample;

$$L_{cons} = d(f_\theta(x), f_\theta(\hat{x}))$$

The perturbation used to generate $\hat{x}$ depends on the variant of consistency regularization used. Virtual adversarial training [Miyato et al., 2017] perturbs a sample in a direction that

maximises the change in prediction, where the direction is determined by computing the gradient of the prediction with respect to the sample. Temporal and self ensembling [Laine and Aila, 2017] use stochastic data augmentation (horizontal flips, translations and Gaussian noise). A variety of distance measures $d(\cdot, \cdot)$ have been used, e.g., squared distance [Laine and Aila, 2017] or cross-entropy [Miyato et al., 2017].

The benefit of the cluster assumption is supported by the formal analysis of Athiwaratkun *et al.* [Athiwaratkun et al., 2019]. They analyse a simplified version of the Π-model [Laine and Aila, 2017] in which perturbation consists of additive Gaussian noise so that $\hat{x} = x + \epsilon h$, where $h \sim \mathcal{N}(0, I)$ and $d(\cdot, \cdot)$ is a squared Euclidean distance. For small constant ϵ, the expected value of the consistency loss term L_{cons} is approximately proportional to the square of the Frobenius norm of the Jacobian $J_{f_\theta}(x)$ of the networks outputs with respect to its inputs:

$$\mathbb{E}[L_{cons}] = \mathbb{E}\left[\left\|f_\theta(x + \epsilon h) - f_\theta(x)\right\|^2\right] \approx \epsilon^2 \left\|J_{f_\theta}(x)\right\|_F^2 . \tag{5.1}$$

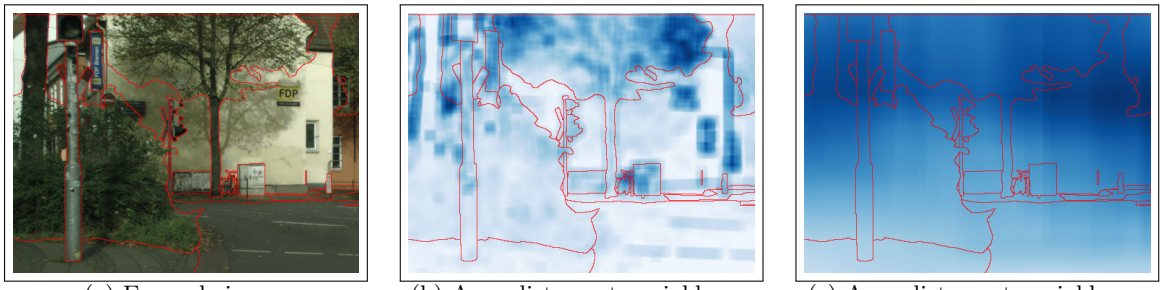
Thus, minimizing L_{cons} directly flattens the decision function in the vicinity of unsupervised samples. Minimizing a supervised term (*e.g.* cross entropy loss) on the other hand will encourage the network to place a decision boundary – and surrounding region of high gradient – between supervised samples that have different classes. Both requirements can be satisfied by moving the decision boundary —and its surrounding region of high gradient— into regions of low sample density.

5.3 Why semi-supervised semantic segmentation appears challenging

We view semantic segmentation as sliding window patch classification with the goal of identifying the class of the patch's central pixel. Given that prior works [Laine and Aila, 2017; Miyato et al., 2017; Sohn et al., 2020] apply perturbations to the raw pixel (input) space our analysis of the data distribution focuses on the raw pixel content of image patches, rather than higher level features from within the network.

We attribute the infrequent success of consistency regularization in natural image semantic segmentation problems to the observations that low density regions in input data do not align well with class boundaries. The presence of such low density regions would manifest as locally larger than average L^2 distances between patches centred on neighbouring pixels that lie either side of a class boundary. In Figure 5.1 we visualise the L^2 distances between neighbouring patches. When using a reasonable receptive field as in Figure 5.1 (c) we can see that the cluster assumption is clearly violated: how much the raw pixel content of the receptive field of one pixel differs from the contents of the receptive field of a neighbouring pixel has little correlation with whether the patches' centre pixels belong to the same class.

To generate the distance maps in Figure 5.1(b, c) we need to compute the L^2 distances between the pixel content of overlapping patches centred on all pairs of horizontally neighbouring pixels. We see two example patches A and B in Figure 5.2(a,b). The first step of computing the L^2 distance is computing $B - A$. Given that each pixel in $B - A$ is the



(a) Example image

(b) Avg. distance to neighbour, patch size 15×15

(c) Avg. distance to neighbour, patch size 225×225

Figure 5.1: In a segmentation task, low-density regions rarely correspond to class boundaries. (a) An image crop from the CITYSCAPES dataset. (b) Average L^2 distance between raw pixel contents of a patch centred at pixel p and four overlapping patches centred on the immediate neighbours of p, using 15×15 pixel patches. (c) Same for a more realistic receptive field size of 225×225 pixels. Dark blue indicates large inter-patch distance and therefore a low density region, white indicates a distance of 0. The red lines indicate segmentation ground truth boundaries.

difference between horizontally neighbouring pixels, it is therefore a patch extracted from the horizontal gradient image $\Delta_{\mathrm{x}} I$ (see Figure 5.2(c)). We wish to compute the square root of the sum of patches extracted from the element-wise squared gradient image $(\Delta_{\mathrm{x}} I)^{\circ 2}$ in a sliding window fashion. This can be achieved by convolving $(\Delta_{\mathrm{x}} I)^{\circ 2}$ with a $H \times W$ box kernel, hence the distance map can be written as $\sqrt{(\Delta_{\mathrm{x}} I)^{\circ 2} * 1^{H \times W}}$.

The lack of variation in the patch-wise distances is easy to explain from a signal processing perspective; the squared gradient image is low-pass filtered by a $H \times W$ box filter. This suppresses the fine details found in the high frequency components of the image, leading to smoothly varying distance map and sample density across the image.

Our analysis of the CITYSCAPES dataset quantifies the challenges involved in placing a decision boundary between two neighbouring pixels that should belong to different classes while generalizing to other images. We find that the L^2 distance between patches centred on neighbouring pixels on either side of a class boundary is $\sim 1/3$ of the distance to the closest patch of the same class found in a different image, as shown in Figure 5.3(a). This finding strongly suggest that the *smoothness assumption* is also violated. We obtained this result by choosing 1000 image patch triplets each consisting of an anchor patch A_i and positive P_i and negative N_i patches with the same and different ground truth classes as A_i respectively. Given that a segmentation model must place a decision boundary between neighbouring pixels of different classes within an image we chose A_i and N_i to be centred on immediately neighbouring pixels on either side of a class boundary. As the model must also generalise from a labelled images to unlabelled images we searched all images except that containing A_i for the P_i belonging to the same class that minimises $|P_i - A_i|^2$. Minimising the distance chooses the best case intra-class distance over which the model must generalise.

The illustration in Figure 5.3(b) visualizes the challenging nature of the problem: the model must learn to place the decision boundary between the patches centred on neighbouring

(a) Patch A (b) Patch B (c) Patch from $\Delta_x I$

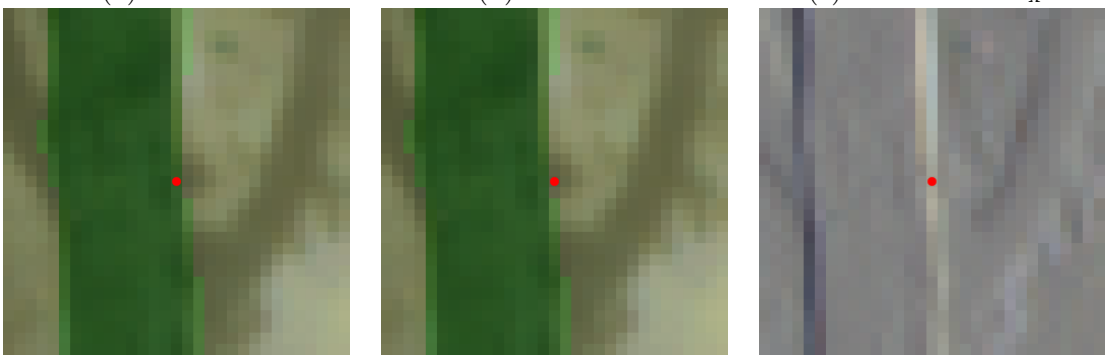


Figure 5.2: (a, b) Two patches centred on horizontally neighbouring pixels, extracted from the Cityscapes Image in Figure 5.1(a). The ground truth vegetation class is overlaid in green. The red dot indicates the central pixel. (c) The same patch extracted from the horizontal gradient image.

pixels, while orienting it sufficiently accurately that it intersects other images at the correct points. Precise positioning and orientation of the decision boundary are essential for good performance.

5.3.1 Consistency regularization without cluster assumption

When considered in the context of our analysis above, the few reports of the successful application of consistency regularization to semantic segmentation – in particular the work of Li et al. [2018b] and Ji et al. [2019] – lead us to conclude that the presence of low density regions separating classes is highly beneficial, but not essential. We therefore suggest an alternative mechanism: that of using non-isotropic natural perturbations such as image augmentation to constrain the orientation of the decision boundary to lie parallel to the directions of perturbation (see the appendix of Athiwaratkun et al. [2019]). We will now explore this using a 2D toy example (please see Appendix A for further implementation details).

Figure 5.4a illustrates the benefit of the cluster assumption with a simple 2D toy semi-supervised mean teacher experiment, in which the cluster assumption holds due to the presence of a gap (low-density region) separating the unsupervised samples into two regions, one for each of the two different classes. The perturbation used for the consistency loss is a simple isotropic Gaussian nudge to both coordinates, and as expected, the learned decision boundary settles neatly between the two clusters.

In Figure 5.4b, the cluster assumption is violated and there are no density differences in the set of unsupervised samples. In this case, the consistency loss does more harm than good; even though it successfully flattens the neighbourhood of the decision function, it does so also across the true class boundary. In order for the consistency regularization to be a net win, it would have to perturb the samples as much as possible, but at the same time avoid crossing the true class boundary.

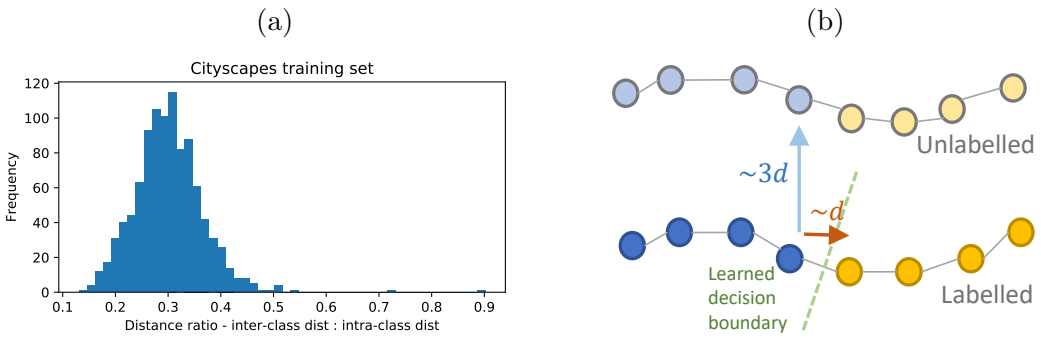
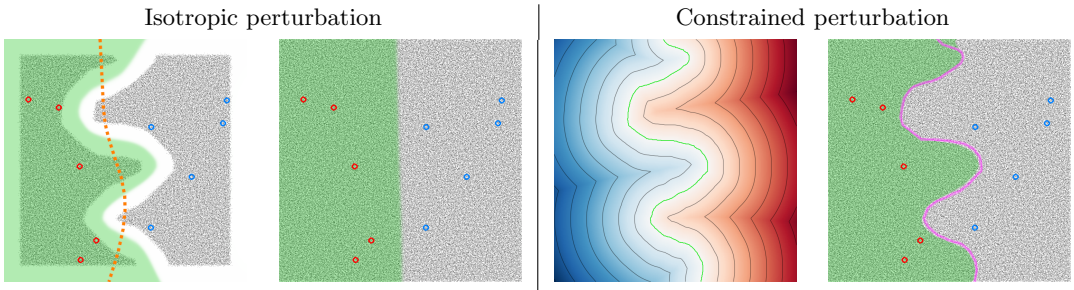


Figure 5.3: (a) histogram of the ratio $|N_i - A_i|^2 / |P_i - A_i|^2$ of the L_2 pixel content inter-class distance between patches A_i and N_i centred on neighbouring pixels either side of class boundary to the intra-class distance between nearest neighbour patches A_i and P_i coming from different images. (b) an illustrative visualization of semantic segmentation sample distribution. The chain of samples (circles) below represents a row of patches from an image changing class (colour) half-way through. The lighter chain above represents an unlabelled image. The dashed green line represents a learned decision boundary. The samples within an image are at a distance of $\sim d$ from one another and $\sim 3d$ from those in another image.

In their appendix, Athiwaratkun et al. [2019] state that consistency regularization with non-isotropic natural perturbations such as image augmentation penalizes the Jacobian norm of the network on the perturbation manifold $\mathcal{M}$ at the position of sample x. L_{cons} only penalizes the network for exhibiting gradient within $\mathcal{M}$ (along the directions of perturbation). This flattens the decision function in directions that lie along $\mathcal{M}$, leaving it free to exhibit gradient perpendicular to it, even in the vicinity of unlabelled samples. This suggests a possible way forward.

In Figure 5.4c, we plot the contours of the distance to the true class boundary, suggesting a potentially better mechanism for perturbation. Indeed, when perturbations lie only along these contours, the probability of crossing the true class boundary is negligible compared to the regularization potential in the remaining dimension. Figure 5.4d shows that the resulting learned decision boundary aligns well with the true class boundary. When low density regions are not present the perturbations must be carefully chosen such that the probability of crossing the class boundary is minimised.

Low-density regions provide an effective signal that can guide consistency regularization by providing areas into which a decision boundary can settle. This illustrative toy example demonstrates an alternative mechanism; the orientation of the decision boundary can be constrained to lie parallel to the directions of perturbation. We therefore argue that consistency regularization can be successful even when the cluster assumption is violated, if the following guidelines are observed: 1) the perturbations must be varied and high-dimensional in order to sufficiently constrain the orientation of the decision boundary in the high-dimensional space of natural imagery, 2) the probability of a perturbation crossing the true class boundary must be very small compared to the amount of exploration in other



(a) Low density region separating classes (b) No low density region (c) Distance map and contours (d) Constrain to dist. map contours

Figure 5.4: Toy 2D semi-supervised classification experiments. Blue and red circles indicate supervised samples from class 0 and 1 respectively. The field of small black dots indicate unsupervised samples. The learned decision function is visualized by rendering the probability of class 1 in green; the soft gradation represents the gradual change in predicted class probability. (a, b) Semi-supervised learning with and without a low density region separating the classes. The dotted orange line in (a) shows the decision boundary obtained with plain supervised learning. (c) Rendering of the distance to the true class boundary with distance map contours. Strong colours indicate greater distance to class boundary. (d) Decision boundary learned when samples are perturbed along distance contours in (c). The magenta line indicates the true class boundary. Section A explains the setup in detail.

dimensions, and 3) the perturbed inputs should be plausible; they should not be grossly outside the manifold of real inputs.

Classic augmentation based perturbations such as cropping, scaling, rotation have a low chance of confusing the output class and have proved to be effective in classifying natural images [Laine and Aila, 2017; Tarvainen and Valpola, 2017]. Given that this approach has positive results in some medical image segmentation problems [Perone and Cohen-Adad, 2018; Li et al., 2018b], it is surprising that it is ineffective for natural imagery. This motivated us to search for stronger and more varied augmentations for semi-supervised semantic segmentation. We later found that the lack of success was due to the network using colour statistics as a short-cut, as we will discuss in further detail in Section 5.5.

5.4 Cutout and CutMix for semi-supervised semantic segmentation

Cutout [DeVries and Taylor, 2017] yielded strong results in semi-supervised classification in UDA [Xie et al., 2019] and FixMatch [Sohn et al., 2020]. The UDA ablation study shows Cutout contributing the lions share of the semi-supervised performance, while the FixMatch ablation shows that Cutout can match the effect of the combination of 14 image operations used by CTAugment. DeVries *et al.* [DeVries and Taylor, 2017] established that Cutout encourages the network to utilise a wider variety of features in order to overcome

the varying combinations of parts of an image being present or masked out. This variety introduced by Cutout suggests that it is a promising candidate for segmentation.

As stated in Section 2.6, CutMix combines Cutout with MixUp, using a rectangular mask to blend input images. Given that MixUp has been successfully used in semi-supervised classification in ICT [Verma et al., 2019] and MixMatch [Berthelot et al., 2019b], we propose using CutMix to blend unsupervised samples and corresponding predictions in a similar fashion.

Our preliminary experiments comparing the Π-model [Laine and Aila, 2017] and the mean teacher model [Tarvainen and Valpola, 2017] indicate that using mean teacher is essential for good performance in semantic segmentation, therefore all the experiments in this chapter use the mean teacher framework. We denote the student network as f_θ and the teacher network as g_ϕ.

5.4.1 Adapting semi-supervised classification algorithms for segmentation

In order to fully assess the performance of CutMix relative to other approaches, we adapted several semi-supervised classification algorithms for segmentation to contrast it against. We will now describe these adaptations.

Standard augmentation

Standard image augmentation has been successfully used as the source of perturbation for semi-supervised classification in a number of prior approaches [Sajjadi et al., 2016b; Laine and Aila, 2017; Tarvainen and Valpola, 2017]. Typically the augmentation consists of geometric transformations such as random crops and flips or affine transformations.

We apply an affine geometric transformation defined by a matrix A to an image x using bi-linear filtering, resulting in the perturbed image $\hat{x} = \text{warp\_affine}(x, A)$. We apply the teacher and student networks to the original and augmented images respectively, resulting in predictions $y = g_\phi(x)$ and $\hat{y} = f_\theta(\hat{x})$. Computing the consistency loss between y and $\hat{y}$ requires that we align them by accounting for the transformation A: the content at pixel $[i, j]$ of image x will be located at $[i, j] \cdot A$ in $\hat{x}$. Therefore, following the approach used by Perone and Cohen-Adad [2018] and Li et al. [2018b] we apply the transformation to the teacher predictions: $L_{cons} = d(\text{bilinear}(y, A), \hat{y})$. So:

$$L_{cons} = \left\| \text{warp\_affine}(g_\phi(x), A) - f_\theta(\text{warp\_affine}(x, A)) \right\|^2 \tag{5.2}$$

At this point we would like to note some of the challenges involved in the implementation. A natural approach would be to use a single system for applying affine transformations, e.g. the affine grid functionality provided by PyTorch [Chintala et al., 2017]; that way both the input images and the predictions can be augmented using the same system. For an apples-to-apples comparison with their results we however wished to exactly match the augmentation system used by Hung et al. [2018] and Mittal et al. [2019a], both of which use functions provided by OpenCV [Bradski, 2000]. This required gathering a precise understanding of

how the relevant functions in OpenCV generate and apply affine transformation matrices in order to match them using PyTorch's affine grid functionality, that must be used to transform predictions.

Interpolation consistency training

ICT [Verma et al., 2019] required almost no adaptation, except that our segmentation networks generate pixel-wise class probability vectors rather than the per-sample class probability vectors generated by a classifier. As in Verma et al. [2019] we draw a per sample (as opposed to per-pixel) blending factor p from the beta distribution $p \sim \beta(\alpha, \alpha)$ where the best value for α was found to be 0.1.

Class probability predictions are generated for two input samples x_a and x_b using the teacher network: $y_a = g_\phi(x_a)$ and $y_b = g_\phi(x_b)$. The images and corresponding predictions are blended using p: $\hat{x}_m = (1-p)x_a + px_b$ and $y_m = (1-p)y_a + py_b$. Student predictions are computed from the mixed image: $\hat{y}_m = f_\theta(\hat{x}_m)$. The consistency loss L_{cons} is computed from y_m and $\hat{y}_m$:

$$L_{cons} = \|y_m - \hat{y}_m\|^2$$

Virtual Adversarial Training

Following the description in Oliver et al. [2018], in a classification scenario VAT starts by drawing a random offset r of the same dimensionality as the input sample x:

$$r \sim \mathcal{N}(0, \frac{\xi}{\sqrt{dim(x)}} I)$$

We compute the adversarial direction r_{adv} as the derivative of the difference in prediction due to r, with respect to r, scaled to length ϵ. We note that we use the teacher g_ϕ network rather than the student f_θ:

$$v = \nabla_r d(g_\phi(x), g_\phi(x+r))$$

$$r_{adv} = \epsilon \frac{v}{||v||}$$

We also scale the adversarial radius ϵ adaptively on a per-image basis by computing it as $s\|\Delta x\|$, where s is a scale factor and $\|\Delta x\|$ is the magnitude of the gradient of the input image. We find that a value of 1 for s works well.

The consistency loss term is computed using the KL-divergences between x and $x + r_{adv}$ (we found that squared difference does not work):

$$L_{cons} = D_{KL}(x, x + r_{adv})$$

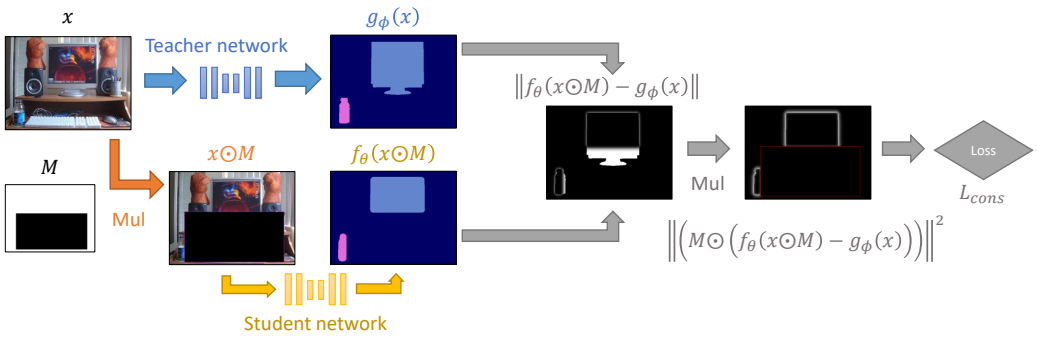


Figure 5.5: Illustration of Cutout regularization for semi-supervised semantic segmentation with the mean teacher framework. Note that we include additional detail in final steps of the computation of L_{cons} in comparison to Figure 5.6 in order to illustrate the masking of the consistency loss.

Cutout

As in DeVries and Taylor [2017] we initialize a mask M with the value 1 and set the pixels inside a randomly chosen rectangle to 0. To apply Cutout in a semantic segmentation task, we mask the input pixels in x with M and disregard the consistency loss for pixels masked to 0 by M. FixMatch [Sohn et al., 2020] uses a *weak* augmentation scheme consisting of crops and flips to predict pseudo-labels used as targets for samples augmented using the *strong* CTAugment scheme. Similarly, we consider Cutout to be a form of *strong* augmentation, so we apply the teacher network g_ϕ to the original image to generate pseudo-targets that are used to train the student f_θ. Using square distance as the metric, we have $L_{cons} = \left\|M \odot (f_\theta(M \odot x) - g_\phi(x))\right\|^2$, where $\odot$ denotes an element-wise product. The computation is illustrated in Figure 5.5.

The original formulation of Cutout [DeVries and Taylor, 2017] for classification used a rectangle of a fixed size and aspect ratio whose centre was positioned randomly, allowing part of the rectangle to lie outside the bounds of the image. For segmentation we obtained better performance with Cutout by randomly choosing the size and aspect ratio and positioning the rectangle so that it lies entirely within the image.

CutMix

CutMix requires two input images that we shall denote x_a and x_b that we mix with the mask M. Following ICT (Verma et al. [2019]) we mix the teacher predictions for the input images $g_\phi(x_a), g_\phi(x_b)$ producing a pseudo target for the student prediction of the mixed image. To simplify the notation, let us define function $mix(a, b, M) = (1 - M) \odot a + M \odot b$ that selects the output pixel based on mask M. We can now write the consistency loss as:

$$L_{cons} = \left|\left|mix\big(g_\phi(x_a), g_\phi(x_b), M\big) - f_\theta\big(mix(x_a, x_b, M)\big)\right|\right|^2. \tag{5.3}$$

The computation is illustrated in Figure 5.6.

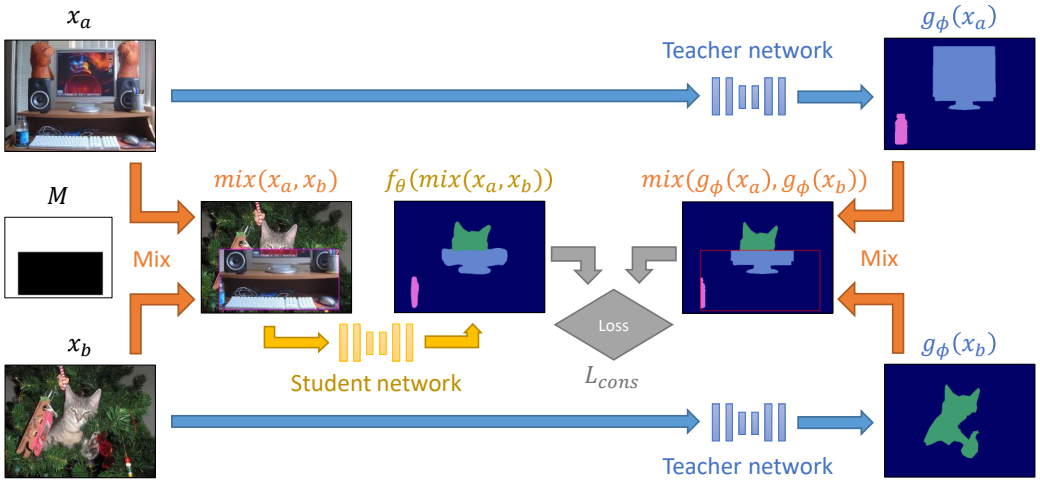


Figure 5.6: Illustration of mixing regularization for semi-supervised semantic segmentation with the mean teacher framework. f_θ and g_ϕ denote the student and teacher networks, respectively. The arbitrary mask M is omitted from the argument list of function *mix* for legibility.

For supervised classification CutMix [Yun et al., 2019] used a rectangle with random size and fixed aspect ratio. The performance of segmentation CutMix was maximized by fixing the area of the rectangle to half that of the image, while varying the aspect ratio and position. Once again the rectangle was positioned so that it lies entirely within the image.

While the augmentations applied by Cutout and CutMix do not appear in real-life imagery, they are reasonable from a visual standpoint. Segmentation networks are frequently trained using image crops rather than full images, so blocking out a section of the image with Cutout can be seen as the inverse operation. Applying CutMix in effect pastes a rectangular region from one image onto another, similarly resulting in a reasonable segmentation task.

5.5 The importance of colour augmentation

The lack of low density regions separating classes described in Section 5.3 offers a convincing explanation for the challenging nature of semi-supervised semantic segmentation. Inspired by the ablation study of Chen et al. [2020a] we will present a simpler alternative. They found that colour statistics can be used by the network as a short-cut for the instance discrimination task used used to train their model. They alleviate this problem by applying colour augmentation, consisting the colour jitter and random greyscale conversion, forcing the network to focus on image content.

Let us consider a semi-supervised semantic segmentation scenario in which we drive consistency regularization with standard augmentation consisting of affine geometric transformation (see Section 5.4.1). The consistency loss term in equation 5.2 applies the affine transformation warp_affine$(\cdot, A)$ in both the student and teacher sides. Minimizing L_{cons} will penalise the network for giving inconsistent class predictions for each individual pixel in

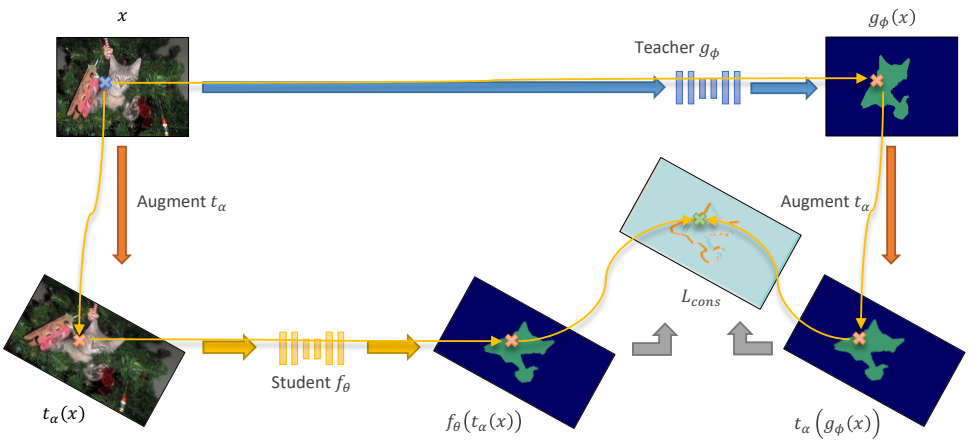


Figure 5.7: Illustration of Mean Teacher unsupervised consistency loss driven by standard augmentation for semantic segmentation problems. The path for a pixel on the neck of the cat leading from the input image x is traced by yellows to the consistency loss map L_{cons} (illustrated prior to computing the mean of the square), with the location of the pixel in each image identified by coloured crosses.

the input image x under geometric augmentation. This is further illustrated in Figure 5.7, in which the yellow arrows follow a single pixel from the input image x through both the student and teacher sides of the consistency loss term. We observe that a potential solution to minimizing L_{cons} is to predict the class of an input pixel using solely its RGB value, ignoring the context provided by surrounding regions of the input image. Instead of using the colour statistics of an image as a short-cut as was observed in the instance discrimination task of Chen et al. [2020a], the network uses the colour of a single pixel as a short-cut in our semantic segmentation task. We propose alleviating this problem by using the same colour augmentation scheme – a colour jitter that alters the brightness, contrast, saturation and hue of an image – employed by the self-supervised approaches of He et al. [2020] and Chen et al. [2020a]. Our experiments in section 5.6 show that the addition of colour augmentation allows standard augmentation and interpolation consistency training to yield improved performance where they otherwise fail to do so. We also observe improved results for all other approaches discussed in Section 5.4.1.

5.6 Experiments

We will now describe our experiments and main results. We will start by describing the training setup and results for the Pascal VOC 2012, Cityscapes data sets. We will follow with the setup and results for ISIC 2017. We compare various perturbation methods in the context of semi-supervised semantic segmentation on Pascal and ISIC.

We use four segmentation networks in our experiments: 1) DeepLab v2 network [Chen et al., 2017] based on ImageNet pre-trained ResNet-101 as used in Mittal et al. [2019a] 2) DeepLab v3+ [Chen et al., 2018] with ImageNet pre-training 3) PSPNet [Zhao et al., 2017]

Architecture	Learning rate
DeepLab v2	3×10^{-5}
DeepLab v3+	1×10^{-5}
DenseNet-161 based Dense U-net	3×10^{-4}
ResNet-101 based PSPNet	1×10^{-4}

Table 5.1: Learning rates used for different architectures. All networks used pre-trained weights for ImageNet classification.

based on ImageNet pre-trained ResNet-101 and 4) Dense U-net [Li et al., 2018a] based on DenseNet-161 [Huang et al., 2017] as used in Li et al. [2018b].

We use cross-entropy for the supervised loss L_{sup} and compute the consistency loss L_{cons} using the Mean teacher algorithm [Tarvainen and Valpola, 2017]. Our final loss term is the weighted sum $L = L_{sup} + \gamma L_{cons}$ where γ is the consistency loss weight.

5.6.1 Pascal VOC 2012 and Cityscapes

Training setup

We use the Adam [Kingma and Ba, 2015] optimization algorithm and we found that different network architectures gave the best performance using different learning rates, presented in Table 5.1. We note that we only used the DeepLab v2 architecture for Cityscapes.

As per the mean teacher algorithm [Tarvainen and Valpola, 2017], after each iteration the weights w_t of the teacher network are updated to be the exponential moving average of the weights w_s of the student: $w_t = \alpha_t w_t + (1 - \alpha_t) w_s$, where $\alpha_t = 0.99$.

The Cityscapes images were down-sampled to half resolution (1024×512) prior to use, as in [Hung et al., 2018]. We extracted 512×256 random crops, applied random horizontal flipping and used a batch size of 4, in keeping with Mittal et al. [2019a].

For the Pascal VOC experiments, we extracted 321×321 random crops, applied a random scale between 0.5 and 1.5 rounded to the nearest 0.1 and applied random horizontal flipping. We used a batch size of 10, in keeping with Hung et al. [2018].

We used a consistency loss weight γ of 1 for both Cutout and CutMix, 0.003 for standard augmentation, 0.01 for ICT and 0.1 for VAT. When using standard augmentation and ICT we had to use low values for the consistency loss weight in order to suppress the effect of the *pixel colour clustering* short-cut (see Section 5.5). Using higher values would typically result in worse accuracy than the supervised baseline. When using colour augmentation we find a consistency loss weight of 1 is optimal for standard augmentation and ICT. For VAT we continue to use a weight of 0.1; we attribute this lower loss weight to the use of KL-divergence in VAT rather than mean squared error for the consistency loss.

We trained for 40,000 iterations for both datasets. We also found that identical hyper-parameters worked well for both.

We used the MIT CSAIL implementation[1] of ResNet-101 based PSPNet [Zhao et al., 2017]. We had to modify[2] their code in order to use our loss functions. We note that we did *not* use the *auxiliary loss* from Zhao et al. [2017], known as the *deep supervision trick* in the MIT CSAIL GitHub repository.

Confidence thresholding

In Chapter 4 we apply confidence thresholding [French et al., 2018b], in which we mask the consistency loss to 0 for samples whose confidence as predicted by the teacher network is below a threshold of 0.968. In the context of segmentation, we found that this masks pixels close to class boundaries as they usually have a low confidence. These regions are often large enough to encompass small objects, preventing learning and degrading performance. Instead we modulate the consistency loss with the proportion of pixels whose confidence is above the threshold. This values grows throughout training, taking the place [Sohn et al., 2020; Ke et al., 2019] of the sigmoidal ramp-up used in Laine and Aila [2017] and Tarvainen and Valpola [2017]. We used a confidence threshold of 0.97 for all segmentation experiments.

Consistency loss with squared error

Most implementations of consistency loss that use squared error (*e.g.* Tarvainen and Valpola [2017]) compute the mean of the squared error over all dimensions. In contrast we sum over the class probability dimension and take the mean over the spatial and batch dimensions. This is more in keeping with the definition of other loss functions used with probability vectors such as cross-entropy and KL-divergence. We also found that this reduces the necessity of scaling the consistency weight with the number of classes; as is required then taking the mean over the class probability dimension [Tarvainen and Valpola, 2017].

5.6.2 Results on Cityscapes and Augmented Pascal VOC

Here we present our results on two natural image datasets and contrast them against the state-of-the-art in semi-supervised semantic segmentation, which is currently the adversarial training approach of Mittal *et al.* Mittal et al. [2019a]. We use two natural image datasets in our experiments. Cityscapes consists of urban scenery and has 2975 images in its training set. Pascal VOC 2012Everingham et al. [2012] is more varied, but includes only 1464 training images, and thus we follow the lead of Hung *et al.* Hung et al. [2018] and augment it using Semantic BoundariesHariharan et al. [2011], resulting in 10582 training images. We adopted the same cropping and augmentation schemes as Mittal et al. [2019a].

In addition to an ImageNet pre-trained DeepLab v2, Hung et al. [2018] and Mittal et al. [2019a] also used a DeepLabV2 network pre-trained for semantic segmentation on the COCO dataset, whose natural image content is similar to that of Pascal. Their results confirm the benefits of task-specific pre-training. Starting from a pre-trained ImageNet

[1] Available at `https://github.com/CSAILVision/semantic-segmentation-pytorch`.

[2] Our modified version can be found in the `logits-from-models` branch of `https://github.com/Britefury/semantic-segmentation-pytorch`.

classifier is representative of practical problems for which a similar segmentation dataset is unavailable for pre-training, so we opted to use these more challenging conditions only.

Our CITYSCAPES results are presented in Table 5.2 as mean intersection-over-union (mIoU) percentages, where higher is better. Our supervised baseline results for CITYSCAPES are similar to those of Mittal et al. [2019a]. We attribute the small differences to training regime choices such as the choice of optimizer. Both the Cutout and CutMix realize improvements over the supervised baseline, with CutMix taking the lead and improving on the adversarial [Hung et al., 2018] and s4GAN [Mittal et al., 2019a] approaches. The addition of colour augmentation results in a slight improvement to the CutOut and CutMix results across the board. We note that CutMix performance is slightly impaired when full size image crops (1024×512) are used getting an mIoU score of $58.75\% \pm 0.75$.

Our PASCAL results are presented in Table 5.3. We will initially discuss our results using the ImageNet pre-trained DeepLab v2 network. Our baselines are considerably weaker than those of Mittal et al. [2019a]; we acknowledge that we were unable to match them. Cutout and CutMix yield improvements over our baseline and CutMix – in spite of the weak baseline – takes the lead, ahead of the adversarial and s4GAN results. Without the use of colour augmentation, standard augmentation and ICT are unable to yield a significant improvement. Virtual adversarial training [Miyato et al., 2017] is able to yield a small improvement. The addition of colour augmentation improves results across the board, with standard augmentation yielding a significant improvement over the baseline, pulling ahead of VAT. ICT comes close to Cutout and CutMix retains the lead. Our results suggest that colour augmentation is necessary to enable standard augmentation and ICT to perform effectively, as without it performance is either reduced in comparison to the baseline or the consistency loss weight must be reduced to the point that consistency regularization has little effect.

For other architectures we compare the baseline with CutMix, with and without colour augmentation. CutMix proves to be effective across all architectures. With the exception of the DenseNet-161 based U-net, colour augmentation generally improves performance.

5.6.3 ISIC 2017 skin lesion segmentation

The ISIC skin lesion segmentation dataset [Codella et al., 2018] consists of dermoscopy images focused on lesions set against skin. It has 2000 images in its training set and is a two-class (skin and lesion) segmentation problem, featuring far less variation than CITYSCAPES and PASCAL.

Training setup

All images were scaled to 248×248 using area interpolation as a pre-process step. Our augmentation scheme consists of random 224×224 crops, flips, rotations and uniform scaling in the range 0.9 to 1.1.

In contrast to Li et al. [2018b] our standard augmentation based experiments allow the samples passing through the teacher and student paths to be arbitrarily rotated and scaled

# labelled	**~1/30 (100)**	**1/8 (372)**	**1/4 (744)**	**All (2975)**
	Results from other work with ImageNet pre-trained DeepLab v2			
Baseline	—	56.2%	60.2%	66.0%
Adversarial	—	57.1%	60.5%	66.2%
s4GAN	—	59.3%	61.9%	65.8%
	Our results: Same ImageNet pre-trained DeepLab v2 network			
Baseline	44.41% ± 0.50	55.25% ± 0.30	60.57% ± 0.51	67.53% ± 0.16
Cutout	47.21% ± 0.78	57.72% ± 0.37	61.96% ± 0.44	67.47% ± 0.30
+ colour aug.	48.28% ± 0.89	58.30% ± 0.33	62.59% ± 0.27	67.93% ± 0.16
CutMix	51.20% ± 1.02	60.34% ± 0.55	63.87% ± 0.32	67.68% ± 0.17
+ colour aug.	**51.98% ± 1.24**	**61.08% ± 0.32**	**64.61% ± 0.25**	**68.11% ± 0.25**

Table 5.2: Performance (mIoU) on CITYSCAPES validation set, presented as *mean* ± *std-err* computed from 5 runs. The results for 'Adversarial' [Hung et al., 2018] and 's4GAN' [Mittal et al., 2019a] are taken from Mittal et al. [2019a].

with respect to one another (within the ranges specified above), where as Li et al. [2018b] use rotations of integer multiples of 90 degrees and flips.

All of our ISIC 2017 experiments use SGD with Nesterov momentum [Sutskever et al., 2013] (momentum value of 0.9) with a learning rate of 0.05 and weight decay of 5×10^{-4}. For Cutout and CutMix we used a consistency weight of 1, for standard augmentation 0.1, for VAT 0.1 and for ICT 0.0005. When using colour augmentation we re-used our weight values that were chosen for the PASCAL dataset; we used a weight of 1 for standard augmentation and ICT and 0.1 for VAT.

We would like to note that scaling the shortest dimension of each image to 248 pixels while preserving aspect ratio reduced performance; the non-uniform scale in the pre-processing step acts as a form of data augmentation.

We apply confidence thresholding and compute squared error consistency loss in the same way was described in Sections 5.6.1 and 5.6.1.

Results

We present our results in Table 5.4. We must first note that our supervised baseline and fully supervised results are noticeably worse that those of Li et al. [2018b]. Given this limitation, we use our results to contrast the effects of the different augmentation schemes used. Our strongest semi-supervised result was obtained using CutMix, followed by standard augmentation, then VAT and Cutout. We found CutMix to be the most reliable, as the other approaches required more hyper-parameter tuning effort to obtain positive results. We were unable to obtain reliable performance from ICT, hence its result is worse than that of the baseline.

While colour augmentation improved the performance of all regularizers on the PASCAL dataset when using the DeepLab v2 architecture, the results for ISIC 2017 are less clear

# labelled	**1/100 (106)**	**1/50 (212)**	**1/20 (529)**	**1/8 (1323)**	**All (10582)**
	Results from other work with ImageNet pretrained DeepLab v2				
Baseline	–	48.3%	56.8%	62.0%	70.7%
Adversarial	–	49.2%	59.1%	64.3%	71.4%
s4GAN+MLMT	–	60.4%	62.9%	67.3%	73.2%
	Our results: Same ImageNet pretrained DeepLab v2 network				
Baseline	33.09%	43.15%	52.05%	60.56%	72.59%
Std. aug.	32.40%	42.81%	53.37%	60.66%	72.24%
+ colour aug.	46.42%	49.97%	57.17%	65.88%	73.21%
VAT	38.81%	48.55%	58.50%	62.93%	72.18%
+ colour aug.	40.05%	49.52%	57.60%	63.05%	72.29%
ICT	35.82%	46.28%	53.17%	59.63%	71.50%
+ colour aug.	49.14%	57.52%	64.06%	66.68%	72.91%
Cutout	48.73%	58.26%	64.37%	66.79%	72.03%
+ colour aug.	52.43%	60.15%	65.78%	67.71%	73.20%
CutMix	**53.79%**	64.81%	66.48%	67.60%	72.54%
+ colour aug.	53.19%	**65.19%**	**67.65%**	**69.08%**	**73.29%**
	Results from other work with ImageNet pre-trained DeepLab v3+				
Baseline	–	unstable	unstable	63.5%	74.6%
s4GAN+MLMT	–	62.6%	66.6%	70.4%	74.7%
	Our results: ImageNet pre-trained DeepLab v3+ network				
Baseline	37.95%	48.35%	59.19%	66.58%	76.70%
CutMix	59.52%	**67.05%**	69.57%	72.45%	76.73%
+ colour aug.	**60.02%**	66.84%	**71.62%**	**72.96%**	**77.67%**
	Our results: ImageNet pre-trained DenseNet-161 based Dense U-net				
Baseline	29.22%	39.92%	50.31%	60.65%	72.30%
CutMix	**54.19%**	**63.81%**	**66.57%**	66.78%	72.02%
+ colour aug.	53.04%	62.67%	63.91%	**67.63%**	**74.16%**
	Our results: ImageNet pre-trained ResNet-101 based PSPNet				
Baseline	36.69%	46.96%	59.02%	66.67%	77.59%
CutMix	**67.20%**	68.80%	73.33%	74.11%	77.42%
+ colour aug.	66.83%	**72.30%**	**74.64%**	**75.40%**	**78.67%**

Table 5.3: Performance (mIoU) on augmented Pascal VOC validation set, using same splits as Mittal *et al.* Mittal et al. [2019a]. The results for 'Adversarial' [Hung et al., 2018] and 's4GAN+MLMT' [Mittal et al., 2019a] are taken from Mittal et al. [2019a].

cut. It harms the performance of VAT and ICT, although we note that we increased the consistency loss weight of ICT to match the value used for Pascal. It yields a noticeable improvement when using standard augmentation and Cutout. Colour augmentation increases the variance of the accuracy when using CutMix, making it slightly less reliable. We hypothesized the the hue jittering component of the colour augmentation may harm

Baseline (50)	Std. aug.	VAT	ICT	Cutout	CutMix	Fully sup. (2000)
Results from Li et al. [2018b] with ImageNet pre-trained DenseUNet-161						
72.85%	**75.31%**	–	–	–	–	79.60%
Our results: Same ImageNet pre-trained DenseUNet-161						
67.64% ±0.82	71.40% ±1.05	69.09% ±0.62	65.45% ±1.57	68.76% ±1.92	**74.57% ±0.46**	78.61% ±0.16
+ colour augmentation						
	73.61% ±1.07	61.94% ±3.01	50.93% ±3.20	73.70% ±1.16	74.51% ±0.87	

Table 5.4: Performance on ISIC 2017 skin lesion segmentation validation set, measured using the Jaccard index (IoU for lesion class). Presented as *mean* ± *std-err* computed from 5 runs. All baseline and semi-supervised results use 50 supervised samples. The fully supervised result ('Full') uses all 2000.

performance in this benchmark, so we tried disabling it when using ICT and VAT. This did not however improve results.

We propose that the good performance of standard augmentation *without colour augmentation* – in contrast to Pascal where it makes barely any difference – is due to unique characteristics of the ISIC 2017 segmentation dataset. The colours present in lesions are darker than those of background skin and the colours present in the dataset are far less varied than in a photographic dataset such as Pascal. While the varied flesh tones and lighting conditions in the dataset would ensure that determining the class of a pixel based solely on its colour would be an ineffective solution, such a local minima could act as an effective step in the path towards a more effective and refined solution.

We initially hypothesized that the strong performance of CutMix on the Cityscapes and Pascal datasets was due to the augmentation in effect 'simulating occlusion', exposing the network to a wider variety of occlusions, thereby improving performance on natural images. This was our motivation for evaluating our approach using the ISIC 2017 lesion segmentation dataset; its images do not feature occlusions and soft edges delineate lesions from skinPerez et al. [2018]. The strong performance of CutMix indicates that the presence of occlusions is not a limiting factor.

5.7 Discussion

In this chapter we have described the conditions necessary – primarily carefully chosen augmentation – for consistency regularization to be a viable solution for semi-supervised semantic segmentation. Given that the data distribution of semantic segmentation lacks low-density regions between classes, our results indicate that the cluster assumption is *not* a pre-requisite for successful semi-supervised learning when using consistency regularization. We have demonstrated two techniques for the successful semi-supervised semantic segmentation: our adapted CutMix regularizer and colour augmentation. Colour augmentation

proved to be effective in photographic datasets such as PASCAL and looks to be a good avenue to explore in these situations. Its value for problems such as skin lesion segmentation is less clear cut, although it improves performance across the board when combined with standard augmentation. We found CutMix to be a robust regularizer, enabling state-of-the-art results and working reliably on natural image datasets, even without colour augmentation. Furthermore, our approach is considerably easier to implement and use than the previous methods based on GAN-style training.

Our finding that the cluster assumption does not hold in semantic segmentation problems combined with our results strongly suggests that the cluster assumption is *not* necessary for successful semi-supervised learning, and that consistency regularization is a very powerful and effective semi-supervised learning algorithm that is applicable to a wide variety of problems. We would advise against focusing on techniques that are designed to exploit the cluster assumption as this limits their application to domains in which this property holds. We hypothesize that other problem domains that involve segmenting continuous signals given sliding-window input – such as audio processing – are likely to exhibit data distributions that are similarly challenging. Our work suggests two avenues for exploration in these domains: mask based regularization; and channel based augmentation that has similar effects as the colour augmentation that we explored here.

We also propose that the challenging nature of the semantic segmentation problem indicates that it is an effective *acid test* for evaluating future semi-supervised or unsupervised regularizers.

We would like to note that in the intervening time since since the completion of this work in early 2020 a number of more recent semi-supervised semantic segmentation approaches have been developed. Yuan et al. [2021] extend RandAugment [Cubuk et al., 2020] for semantic segmentation and use a two-stage teacher student training regime, with a custom loss term and separate batch normalization statistics. Ke et al. [2022] drive consistency regularization with RandAugment [Cubuk et al., 2020] and Cutout DeVries and Taylor [2017] and use a three stage training regime. The Classmix approach of Olsson et al. [2021] uses a custom mixing based data augmentation strategy – similar to the one discussed in this chapter – that derives the mixing mask from network predictions (instead of from rectangles). They select a subset of the classes present in a predicted segmentation map and use their union as the mixing mask.

Since the introduction of Mask R-CNN [He et al., 2017] we abandoned semantic segmentation and contour detection for the instance segmentation component of our fisheries surveillance processing pipeline. While semantic segmentation is therefore not of practical use in this specific application, we consider that value of the work presented in this chapter lies in our analysis of the problem and our exploration of regularizers. The challenging nature of semantic segmentation leads us to believe that a semi-supervised regularizer – namely CutMix – that is able to yield positive results is likely to be applicable elsewhere.

In the next chapter we continue the theme of mask-based semi-supervised learning, developing an effective masked based regularizer for semi-supervised image classification. We

continue to use the Mean Teacher algorithm [Tarvainen and Valpola, 2017] and the confidence thresholding technique presented in the previous chapter.

6 Milking CowMask for semi-supervised image classification

In this chapter we contribute a mask-based regularizer for semi-supervised image classification that we presented in our paper French. et al. [2022]. Following the theme established in Chapters 4 and 5 we use the Mean Teacher [Tarvainen and Valpola, 2017] variant of consistency regularization with confidence thresholding, driven by data augmentation [Laine and Aila, 2017; Oliver et al., 2018].

Mask based augmentation schemes have proven to be particularly effective for semi-supervised learning. As noted in Section 5.4, the augmentation schemes used in UDA [Xie et al., 2019] and FixMatch [Sohn et al., 2020] employ a combination of the mask-based Cutout [DeVries and Taylor, 2017] and RandAugment [Cubuk et al., 2020] or CTAugment [Berthelot et al., 2019a]. The UDA ablation study shows Cutout contributing the lions share of the semi-supervised performance, while the FixMatch ablation shows that Cutout can match the effect of the combination of 14 image operations used by CTAugment.

We introduce a simple masking strategy that we call CowMask, whose shapes and appearance are more varied than the rectangular masks used by CutOut and RandErase [Zhong et al., 2020]. When used to erase parts of an image in a similar fashion to RandErase, CowMask outperforms rectangular masks in the majority of semi-supervised image classifications tasks that we tested.

We extend the Interpolation Consistency Training (ICT) algorithm [Verma et al., 2019] to use mask-based mixing, using both rectangular masks as in CutMix [Yun et al., 2019] and CowMask. Both CutMix and CowMask exhibit strong semi-supervised learning performance, with CowMask outperforming rectangular mask based mixing in the majority of cases.

CowMask based mixing achieves competitive semi-supervised image classification results on ImageNet and on multiple small image datasets, without the use of multi-stage training procedures or complex training objectives. Our 10% semi-supervised ImageNet results were state-of-the-art when we produced them in mid-January 2020. They were however beaten by the results of SimCLR [Chen et al., 2020a] one month later.

Our implementation is available at `https://github.com/google-research/google-research/tree/master/milking_cowmask`.

6.1 Subsequent work

The results presented in this chapter were obtained in January 2020. A number of approaches have been presented in the intervening period, a number of which yield results superior to the ones we present. We will mention them briefly here.

Recent self-supervised methods – namely SimCLR [Chen et al., 2020a] and TWIST [Wang et al., 2021] – have yielded strong semi-supervised classification results in a two step method consisting of self-supervised pre-training followed by supervised fine-tuning using the labelled subset of the training set. CoMatch [Li et al., 2021] combines consistency regularization with self-supervised contrastive learning and is discussed in further detail in Section 2.7.4. Meta Pseudo Labels [Pham et al., 2021] combines pseudo labelling – in which a teacher network predicts labels used to train a student – with meta-learning objectives that use the performance of the student on supervised samples to guide the training of the teacher.

6.2 CowMask

Here, we propose CowMask; a simple approach to generating flexibly shaped masks, so called due to its Friesian cow-like appearance. Example CowMasks are shown in Figure 6.1.

We note that the concurrent work FMix [Harris et al., 2020] uses an inverse Fourier transform to generate masks with a similar visual appearance.

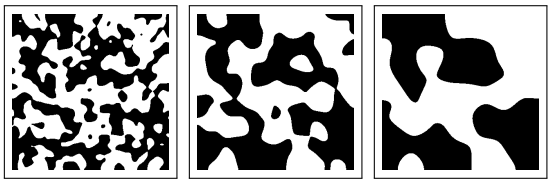


Figure 6.1: Example CowMasks with $p = 0.5$ and varying σ.

Briefly, a CowMask is generated by applying Gaussian filtering of scale σ to normally distributed noise. A threshold τ is chosen such that a proportion p of the smooth noise pixels are below τ. Pixels with a value below τ are assigned a value of 1, or 0 otherwise. The scale of the mask features is controlled by σ – as seen in the examples in Figure 6.1 – and is drawn from a log-uniform distribution in the range $(\sigma_{min}, \sigma_{max})$. The proportion p of pixels with a value of 1 is drawn from a uniform distribution in the range (p_{min}, p_{max}). The procedure for generating a CowMask is provided in Algorithm 1.

6.3 Semi-Supervised Learning Method

We adopt the Mean Teacher [Tarvainen and Valpola, 2017] framework as the basis of our approach. We use two networks; the student $f_\theta(\cdot)$ and the teacher $g_\phi(\cdot)$, both of which predict class probability vectors. The student is trained by gradient descent as normal. After

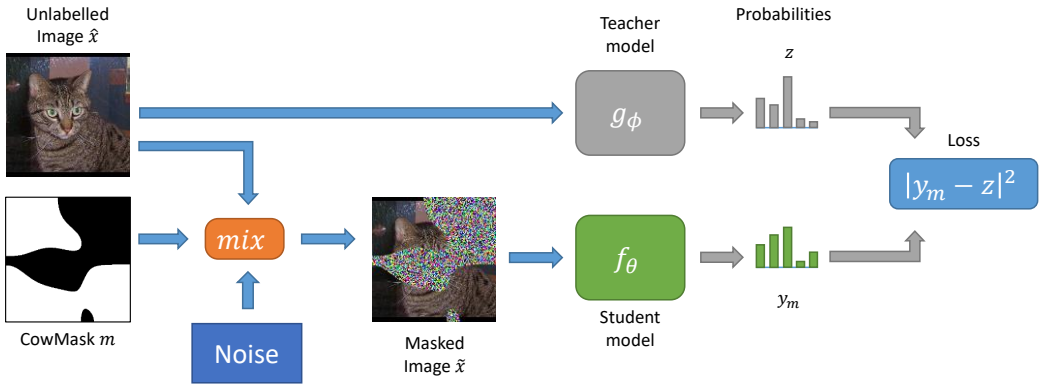


Figure 6.2: Illustration of the unsupervised mask based erasure consistency loss component of semi-supervised image classification. Blue arrows carry image or mask content and grey arrows carry probability vectors. Note that confidence thresholding is not illustrated here.

Algorithm 1 CowMask generation algorithm. See Figure 6.1 for example output.

Require: mask size $H \times W$
Require: scale range $(\sigma_{min}, \sigma_{max})$
Require: proportion range (p_{min}, p_{max})
Require: inverse error function erf^{-1}

$\sigma \sim \log\mathcal{U}(\sigma_{min}, \sigma_{max})$ ▷ Randomly choose sigma
$p \sim \mathcal{U}(p_{min}, p_{max})$ ▷ Randomly choose proportion
$\mathbf{x} \sim \mathcal{N}^{H \times W}(0, 1)$ ▷ Per-pixel Gaussian noise
$\mathbf{x}_s = \text{gaussian\_filter\_2d}(x, \sigma)$ ▷ Filter noise
$m = \text{mean}(\mathbf{x}_s)$ ▷ Compute mean and std-dev
$s = \text{std\_dev}(\mathbf{x}_s)$
$\tau = m + \sqrt{2} \cdot \text{erf}^{-1}(2p - 1) \cdot s$ ▷ Compute threshold
$\mathbf{c} = \mathbf{x}_s \leq \tau$ ▷ Threshold filtered noise
Return c

every update to the student, the weights of the teacher are updated to be an exponential moving average of those of the student using $\phi' = \phi\alpha + \theta(1 - \alpha)$. The momentum α controls the trade-off between the stability and the speed at which the teacher follows the student.

Our training set consists of a set of supervised samples S consisting of input images s and corresponding target labels t, and a set of unsupervised samples U consisting only of input images u. Given a labelled dataset we select the supervised subset randomly such that it maintains the class balance of the overall dataset[1] as is standard practice in the literature. All available samples are used as unsupervised samples. Our models f_θ are then trained to

[1] We use `StratifiedShuffleSplit` from Scikit-Learn [Buitinck et al., 2013]

minimize a combined loss:

$$L = L_{\mathcal{S}}(f_\theta(s), t) + \omega L_{\mathcal{U}}(f_\theta(u), g_\phi(u))$$

where we use standard cross entropy loss for the supervised loss $L_{\mathcal{S}}(\cdot)$ and consistency loss for the unsupervised loss $L_{\mathcal{U}}(\cdot)$ that is modulated by the unsupervised loss weight ω.

We explore two different types of mask-based consistency regularization: *mask-based erasure* and *mask-based mixing*. In mask-based erasure we perturb our input data by erasing the part of the input image corresponding to a randomly sampled mask. In mask-based mixing we blend two input images together, with the blending weights given by the sampled mask. We follow the nomenclature of Cutout and CutMix, using the terms CowOut and CowMix to refer to CowMask based erasure and mixing respectively.

6.3.1 Mask-based Augmentation by Erasure

Mask-based erasure can function as an augmentation that can be added to the standard augmentation scheme used for the dataset at hand, with one caveat. Similar to prior work [Xie et al., 2019; Berthelot et al., 2019a; Sohn et al., 2020] we found it necessary to split our augmentation into a 'weak' standard augmentation scheme (e.g. crop and flip) and a 'strong' rich scheme; RandAugment in the case of the prior works mentioned or CowOut in our work. Weakly augmented samples are passed to the teacher network, generating predictions that are used as pseudo-targets that the student is encouraged to match for strongly augmented variants of the same samples. Using 'strong' erasure augmentation to generate pseudo-targets resulted in unstable training.

The π-model [Laine and Aila, 2017] and the Mean Teacher model [Tarvainen and Valpola, 2017] both use a Gaussian ramp-up function to modulate the effect of consistency loss during the early stages of training. Reinforcing the random predictions of an untrained network was found to harm performance. In place of a ramp-up we opt to use confidence thresholding (see Section 4.1.2 and see French et al. [2018b]). Consistency loss is masked to zero for samples for which the teacher networks' predictions are below a specified threshold. FixMatch [Sohn et al., 2020] uses confidence thresholding for similar reasons.

Our procedure for computing unsupervised consistency loss based on erasure is provided in Algorithm 2 and is illustrated in Figure 6.2. For our small image experiments we found that the best value for the unsupervised weight factor ω is 1.

6.3.2 Mask-based Mixing

Alternatively, we can construct an unsupervised consistency loss by mask-based *mixing* of images in place of erasure. Our approach for mixing image pairs using masks is essentially that of Interpolation Consistency Training (ICT) [Verma et al., 2019]. ICT works by passing the original image pair to the teacher network, the blended image to the student and encourages the students prediction to match the blended teacher predictions. Where ICT draws per-pair blending factors a beta distribution, we mix images using a mask, and mix probability predictions with the mean of the mask (the proportion of pixels with a value of 1).

Algorithm 2 CowOut: erasure-based unsupervised loss.

Require: unlabelled image $\mathbf{x}$, CowMask $\mathbf{m}$
Require: teacher model g_ϕ
Require: student model f_θ
Require: confidence threshold ψ
$\hat{\mathbf{x}} = \text{std\_aug}(\mathbf{x})$ ▷ standard augmentation
$\mathbf{z} = \text{stop\_gradient}(g_\phi(\hat{\mathbf{x}}))$ ▷ teacher pred.
$q = \max_i \mathbf{z}[i] \geq \psi$ ▷ confidence mask
$\epsilon \sim N(0, I)$ ▷ generate noise image
$\hat{\mathbf{x}}_m = \hat{\mathbf{x}} * \mathbf{m} + \epsilon * (1 - \mathbf{m})$ ▷ apply mask
$\mathbf{y}_m = f_\theta(\hat{\mathbf{x}}_m)$ ▷ student prediction
$d = q * ||\mathbf{y}_m - \mathbf{z}||_2^2$ ▷ cons. loss
Return d

Confidence thresholding required adaptation for use with mix-based regularization. Rather than applying confidence thresholding to the blended teacher probability predictions we opted to blend the confidence values before thresholding as this gave slightly better results. Further improvements resulted from modulating the consistency loss by the proportion of samples in the batch whose predictions cross the confidence threshold, rather masking the loss for each sample individually.

The procedure for computing unsupervised mix consistency loss is provided in Algorithm 3 and illustrated in Figure 6.3. We found that a higher weight ω was appropriate for mix consistency loss; we used a value of 30 for our small image experiments.

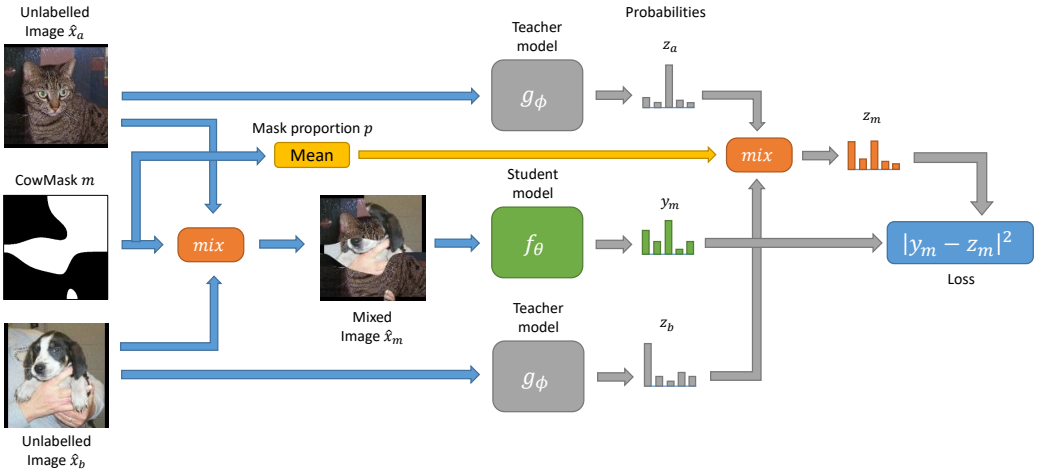


Figure 6.3: Illustration of the unsupervised masked based mixing loss component of semi-supervised image classification. Blue arrows carry image or mask content, grey arrows carry probability vectors and yellow carry scalars. Please note that confidence thresholding is not illustrated here.

Algorithm 3 CowMix: mixing-based unsupervised loss.

Require: unlabelled images $\mathbf{x}_a$, $\mathbf{x}_b$
Require: CowMask $\mathbf{m}$
Require: teacher model g_ϕ
Require: student model f_θ
Require: confidence threshold ψ

$\hat{\mathbf{x}}_a = \text{std\_aug}(\mathbf{x}_a)$ ▷ standard augmentation
$\hat{\mathbf{x}}_b = \text{std\_aug}(\mathbf{x}_b)$
$\mathbf{z}_a = \text{stop\_gradient}(g_\phi(\hat{\mathbf{x}}_a))$ ▷ teacher pred.
$\mathbf{z}_b = \text{stop\_gradient}(g_\phi(\hat{\mathbf{x}}_b))$
$c_a = \max_i \mathbf{z}_a[i]$ ▷ confidence of prediction
$c_b = \max_i \mathbf{z}_b[i]$
$\hat{\mathbf{x}}_m = \hat{\mathbf{x}}_a * \mathbf{m} + \hat{\mathbf{x}}_b * (1 - \mathbf{m})$ ▷ mix images
$p = \text{mean}(\mathbf{m})$ ▷ scalar mean of mask
$\mathbf{z}_m = \mathbf{z}_a * p + \mathbf{z}_b * (1 - p)$ ▷ mix tea. preds.
$c_m = c_a * p + c_b * (1 - p)$ ▷ mix confidences
$q = \text{mean}(c_m \geq \psi)$ ▷ mean of conf. mask
$\mathbf{y}_m = f_\theta(\hat{\mathbf{x}}_m)$ ▷ stu. pred. on mixed image
$d = q||\mathbf{y}_m - \mathbf{z}_m||_2^2$ ▷ cons. loss
Return d

Approach	Architecture	Params.	Top-5 err.	Top-1 err.
	Our baselines			
Sup 10%	ResNet-152	60M	22.12%	42.91%
Sup 100%	ResNet-152	60M	5.67%	21.33%
	Other work: self-supervised pre-training then fine-tune			
SimCLR[1]	ResNet-50	24M	12.2%	34.4%
SimCLR	ResNet-50×2	94M	8.8%	28.3%
SimCLR	ResNet-50×4	375M	7.4%	25.6%
TWIST[2]	ResNet-50	24M	9.0%.	28.3%
TWIST	ResNet-50×2	94M	**7.2%**	**24.7%**
	Other work: semi-supervised			
Mean Teacher[3]	ResNeXt-152	62M	$9.11\% \pm 0.08\%$	–
UDA[4]	ResNet-50	24M	11.2%	31.22%
FixMatch[5]	ResNet-50	24M	$10.87 \pm 0.24\%$	$28.54 \pm 0.32\%$
S^4L Full (MOAM)[6]	ResNet-50×4	375M	8.77%	26.79%
CoMatch[7]	ResNet-50	24M	8.4%	26.4%
Meta Pseudo Labels[8]	ResNet-50	24M	8.62%	26.11%
	Our results			
CowMix	ResNet-152	60M	$8.76 \pm 0.03\%$	$26.06 \pm 0.08\%$

[1] Chen et al. [2020a], [2] Wang et al. [2021], [3] Tarvainen and Valpola [2017], [4] Xie et al. [2019], [5] Sohn et al. [2020], [6] Zhai et al. [2019], [7] Li et al. [2021], [8] Pham et al. [2021]

Table 6.1: Results on ImageNet with 10% labels. Note that S^4L involves three steps with different training procedures, while CowMix involves a single training run. SimCLR is able to beat CowMix, but only when using a very large model.

6.4 Experiments and results

We first evaluate CowMix for semi-supervised consistency regularization on the challenging ImageNet dataset, where we are competitive with the state of the art. Next, we examine CowOut and CowMix further and compare with previously proposed methods by trying multiple versions of our approach combined with multiple models on three small image datasets: CIFAR-10, CIFAR-100 and SVHN. The training regimes used for both ImageNet and the small image datasets are sufficiently similar that we used the same codebase for all of our experiments.

Our results are obtained by using the teacher network for evaluation. We report our results as error rates presented as the mean ± 1 standard error computed from the results of 5 runs, each of which uses a different subset of samples as the supervised set. Supervised sets are consistent for all experiments for a given dataset and number of supervised samples.

6.4.1 ImageNet 2012

We contrast the following scenarios: a supervised baseline using 10% of the dataset, semi-supervised training with the same 10% of labelled examples using CowMix consistency regularization on all unlabelled examples, and fully supervised training with all 100% labels.

Setup

We used the ResNet-152 architecture. We adopted a training regime as similar as possible to a standard ImageNet ResNet training protocol. We used a batch size of 1024 and SGD with Nesterov Momentum [Sutskever et al., 2013] set to 0.9 and weight decay (via L2 regularization) set to 0.00025. Our standard augmentation scheme consists of inception crop, random horizontal flip and colour jitter, as in Tarvainen and Valpola [2017].We found that the standard learning rate of 0.1 resulted in unstable training, but were able to stabilise it by reducing the learning rate to 0.04 [Tarvainen and Valpola, 2017]. We found that our approach benefits from training for longer than in supervised settings, so we doubled the number of training epochs to 180 and stretched the learning rate schedule by a factor of 2, reducing the learning rate at epochs 60, 120 and 160 and reduced it by a factor of 0.2 rather than 0.1. We used a teacher EMA momentum α of 0.999.

We obtained our CowMix results using a mix loss weight of 100 and a confidence threshold of 0.5. We drew the CowMask σ scale parameter from the range $(32, 128)$.

Results

Our ImageNet results are presented in Table 6.1. The CoMatch [Li et al., 2021] and Meta Pseudo Labels [Pham et al., 2021] approaches (both more recent than our CowMix work) uses the smaller ResNet-50 architecture and are able beat our top-5 error result and are slightly behind our top-1 error result. We match the S^4L MOAM [Zhai et al., 2019] top-5 error result and beat their top-1 error result, with a simple end-to-end approach and a significantly smaller model. By comparison the S^4L MOAM result is obtained using a 3-stage training and fine-tuning procedure. Recent self-supervised approaches have achieved impressive semi-supervised results on ImageNet by first training a model self-supervised fashion followed by fine-tuning using a subset of the labelled data. The recent SimCLR [Chen et al., 2020a] approach (concurrent work) beats our result when using a much larger model. The more recent TWIST [Wang et al., 2021] approach beats our result using a double-width ResNet-50 that has only 50% more parameters than the ResNet-152 that we use. We tested our approach with wider models (e.g. ResNet-50×2) but obtained our best results from the deeper and commonly used ResNet-152.

6.4.2 Small Image Experiments

Alongside CowOut and CowMix we implemented and evaluated Mean Teacher, CutOut/RandErase and CutMix, and we compare our method against these using the CIFAR-10, CIFAR-100, and SVHN datasets.

We note the following differences between our implementation and those of CutOut and CutMix: 1. Our boxes are chosen so that they entirely fit within the bounds of the mask, whereas CutOut and CutMix use a fixed or random size respectively and centre the box anywhere within the mask, with some of the box potentially being outside the bounds of the mask. 2. CutOut uses a fixed size box, CutMix randomly chooses an area but constrains the aspect ratio to be that of the mask, we choose both randomly.

Setup

For the small image experiments we use a 27M parameter Wide ResNet 28-96x2d with shake-shake regularization [Gastaldi, 2017]. We note that as a result of a mistake in our implementation we used a 3×3 convolution rather than a 1×1 in the residual shortcut connections that either down-sample or change filter counts, resulting in a slightly higher parameter count.

The standard Wide ResNet training regime [Zagoruyko and Komodakis, 2016] is very similar to that used for ImageNet. We used the optimizer, but with weight decay of 0.0005 and a batch size of 256. As before, the standard learning rate of 0.1 had to be reduced to ensure stability, this time to 0.05. The small image experiments also benefit from training for longer; 300 epochs instead of the standard 200 used in supervised settings. The adaptations made to the Wide ResNet learning rate schedule were nearly identical to those made to the ImageNet schedule. We doubled its length and reduced the learning rate by a factor of 0.2 rather than 0.1. We did however remove the last step; the learning rate is reduced at epochs 120 and 240 rather than epochs 60, 120 and 160 as used in supervised settings. For erasure experiments we used a teacher EMA momentum α of 0.99 and for mixing experiments we used 0.97.

When using CowOut and CowMix we obtained the best results when the CowMask scale parameter σ is drawn from the range $(4, 16)$. We note that this corresponds to a range of $(\frac{1}{8}, \frac{1}{2})$ relative to the 32×32 image size and that the σ range used in our ImageNet experiments bears a nearly identical relationship to the 224×224 image size used there. For erasure experiments using CowOut we obtained the best results when drawing p; the proportion of pixels that are retained from the range $(0.25, 1)$. Intuitively it makes sense to retain at least 25% of the image pixels as encouraging the network to predict the same result for an image and a blank space is unlikely to be useful. For mixing experiments using CowMix we obtained the best results when drawing p from the range $(0.2, 0.8)$.

We performed hyper-parameter tuning on the CIFAR-10 dataset using 1,000 supervised samples and evaluating on 5,000 training samples held out as a validation set. The best hyper-parameters found were used as-is for CIFAR-100 and SVHN.

Results

Our results for CIFAR-10, CIFAR-100 and SVHN datasets are presented in Tables 6.2, 6.4 and 6.3 respectively. Considering the techniques we explore we find that mix-based regularization outperforms erasure based regularization, irrespective of the mask generation method used.

We would like to note that our 27M parameter model is larger than the 1.5M parameter models used for the majority of results in other works, so we cannot make an apples-to-apples comparison in these cases. Our CIFAR-10 results are competitive with recent work, except in small data regimes of less than 500 samples where EnAET [Wang et al., 2019b] and FixMatch [Sohn et al., 2020] outperform CowMix. Our CIFAR-100 and SVHN results are competitive with recent approaches but are not state of the art. We note that we did not tune our hyper-parameters for these datasets.

6.5 Effectiveness of CowMix

We explain the effectiveness of CowMix by considering the effects of CowMask and mixing based semi-supervised learning separately.

DeVries and Taylor [2017] established that Cutout – that uses a box shaped mask similar to RandErase - encourages the network to utilise a wider variety of features in order to overcome the varying combinations of parts of an image being present or masked out. In comparison to a rectangular mask the more flexibly shaped CowMask provides more variety and has less correlation between regions of the mask. Increasing the range of combinations of image regions being left intact or erased enhances its effectiveness.

The MixUp [Zhang et al., 2018] and CutMix [Yun et al., 2019] regularizers demonstrated that encouraging network predictions vary smoothly between two images as they are mixed – using either interpolation or mask-based mixing – improved supervised performance, with mask-based mixing offering the biggest gains. We adapted CutMix – in a similar fashion to ICT – for semi-supervised learning and showed that mask based mixing yields significant gains when used as an unsupervised regularizer. CowMix adds the benefits of flexibly shaped masks into the mix.

6.6 Discussion

In this chapter we presented and evaluated *CowMask* for use in semi-supervised consistency regularization, achieving a result competitive with the state of the art on semi-supervised ImageNet, with a much simpler method than in previously proposed approaches, using standard networks and training procedures. We examined both erasure-based and mixing-based augmentation using CowMask, and find that the mix-based variant – which we call *CowMix* – is particularly effective for semi-supervised learning. Further experiments on small image data sets SVHN, CIFAR-10, and CIFAR-100 demonstrate that CowMask is widely applicable.

In the context of the application of by-catch quantification, species identification by image classification is an important component. A successful semi-supervised image classification approach offers a potential solution to the annotation bottleneck that afflicts CatchMonitor. The approach developed in this chapter was explored and in part employed in the final system, as will be discussed in Chapter 7.

Labeled samples	**40**	**50**	**100**	**250**	**500** ⋯
			Other work: uses smaller Wide ResNet 28-2 model with 1.5M parameters		
EnAET		**16.45%**	**9.35%**	7.6% ± 0.29	7.27%
UDA				8.76% ± 0.28	6.68% ± 0.08
MixMatch				11.08% ± 0.42	9.65% ± 0.44
ReMixMatch	14.98% ± 0.82			6.27% ± 0.26	
FixMatch (RA)	**13.81% ± 0.82**			**5.07% ± 0.36**	
			Other work: uses 26M parameter models		
EnAET					
UDA					
MixMatch					
			Our results: uses 27M parameter Wide ResNet 28-96x2d with shake-shake		
Supervised		76.01% ± 0.68	69.74% ± 0.93	58.41% ± 0.72	47.12% ± 0.80
	Augmentation / erasure based regularization				
Mean teacher		75.68% ± 1.66	67.77% ± 1.86	47.95% ± 2.02	29.72% ± 2.57
RandErase		74.67% ± 0.95	62.86% ± 1.61	37.63% ± 3.22	19.22% ± 1.49
CowOut		72.55% ± 1.70	56.72% ± 1.74	28.45% ± 3.14	14.00% ± 0.82
	Mix based regularization				
ICT		80.08% ± 1.15	72.96% ± 1.99	44.92% ± 3.51	17.10% ± 0.96
CutMix		66.06% ± 7.07	34.05% ± 2.77	9.01% ± 1.61	6.81% ± 0.47
CowMix		**55.46% ± 6.81**	23.00% ± 1.77	7.56% ± 0.42	**5.34% ± 0.36**

Labeled samples	**1000**	**2000**	**4000**	**ALL**
			Other work: uses smaller Wide ResNet 28-2 model with 1.5M parameters	
EnAET	6.95%	6.0%	5.35%	
UDA	5.87% ± 0.04	5.51% ± 0.07	5.29% ± 0.08	
MixMatch	7.75% ± 0.25	7.03% ± 0.17	6.24% ± 0.11	
ReMixMatch	5.73% ± 0.18		5.14% ± 0.09	
FixMatch (RA)			4.26% ± 0.10	
			Other work: uses 26M parameter models	
EnAET			4.18% ± 0.1	**1.99%**
UDA			3.7% / **2.7%**	
MixMatch			4.95% ± 0.13	
			Our results: uses 27M parameter Wide ResNet 28-96x2d with shake-shake	
Supervised	36.61% ± 0.50	24.53% ± 0.36	14.81% ± 0.19	3.57% ± 0.04
	Augmentation / erasure based regularization			
Mean teacher	14.14% ± 0.25	8.79% ± 0.07	6.92% ± 0.07	3.04% ± 0.03
RandErase	11.87% ± 0.33	7.05% ± 0.06	5.27% ± 0.08	2.59% ± 0.04
CowOut	8.98% ± 0.50	6.27% ± 0.18	4.97% ± 0.05	2.50% ± 0.04
	Mix based regularization			
ICT	10.40% ± 0.28	7.75% ± 0.55	5.97% ± 0.05	3.45% ± 0.03
CutMix	5.44% ± 0.17	4.62% ± 0.07	4.11% ± 0.08	2.78% ± 0.06
CowMix	**4.73% ± 0.17**	**4.13% ± 0.07**	3.61% ± 0.03	2.56% ± 0.03

Table 6.2: Results on CIFAR-10 test set, error rates as *mean* ± *std-err* of 5 independent runs.

Labeled samples	**40**	**100**	**250**	**500** ⋯
Other work: uses smaller Wide ResNet 28-2 model with 1.5M parameters				
EnAET		16.92%	**3.21% ± 0.23**	**3.05%**
UDA				
MixMatch			3.78% ± 0.23	3.64% ± 0.30
ReMixMatch	**3.55% ± 0.88**	3.10% ± 0.32		2.83% ± 0.25
FixMatch (RA)	3.96% ± 0.66		**2.48% ± 0.28**	
Other work: uses 26M parameter models				
EnAET				
Our results: uses 27M parameter Wide ResNet 28-96x2d with shake-shake				
Supervised		71.24% ± 2.41	37.02% ± 2.75	18.85% ± 0.67
Augmentation / erasure based regularization				
Mean teacher		62.16% ± 4.88	8.23% ± 2.07	3.84% ± 0.07
RandErase		52.55% ± 9.85	7.61% ± 0.76	6.17% ± 0.56
CowOut		66.66% ± 8.81	12.11% ± 0.81	5.94% ± 0.17
Mix based regularization				
CutMix		**9.54% ± 1.13**	5.62% ± 0.42	4.32% ± 0.23
CowMix		9.73% ± 1.79	3.59% ± 0.13	3.80% ± 0.14

Labeled samples	**1000**	**2000**	**4000**	**ALL**
Other work: uses smaller Wide ResNet 28-2 model with 1.5M parameters				
EnAET	2.92%	**2.84%**	2.69%	
UDA	2.55% ± 0.31			
MixMatch	3.27% ± 0.25	3.04% ± 0.16	2.89% ± 0.11	
ReMixMatch		**2.42% ± 0.13**		
FixMatch (RA)	**2.28% ± 0.15**			
Other work: uses 26M parameter models				
EnAET	2.42%			
Our results: uses 27M parameter Wide ResNet 28-96x2d with shake-shake				
Supervised	11.71% ± 0.25	8.23% ± 0.17	6.01% ± 0.21	2.82% ± 0.04
Augmentation / erasure based regularization				
Mean teacher	3.75% ± 0.04	3.61% ± 0.07	3.47% ± 0.05	2.73% ± 0.02
RandErase	4.81% ± 0.21	3.66% ± 0.07	3.21% ± 0.10	2.36% ± 0.02
CowOut	4.36% ± 0.13	3.59% ± 0.11	3.04% ± 0.02	2.42% ± 0.04
Mix based regularization				
CutMix	3.79% ± 0.18	3.26% ± 0.12	2.92% ± 0.04	2.29% ± 0.04
CowMix	3.72% ± 0.27	3.13% ± 0.05	2.90% ± 0.08	2.18% ± 0.03

Table 6.3: Results on SVHN test set, error rates as *mean* ± *std-err* of 5 independent runs.

# Labels	1000	5000	10000	ALL
Other work: uses 1.5M parameters Wide ResNet 28-2				
EnAET	58.73%	31.83%	26.93% ± 0.23	20.55%
MixMatch			25.88% ± 0.25	
FixMatch			**22.60% ± 0.16**	
Other work: uses 26M parameter models				
EnAET			22.92%	16.87%
Our results: 27M param WRN 28-96x2d				
Supervised	78.80% ± 0.10	49.24% ± 0.18	36.04% ± 0.12	18.82% ± 0.10
Augmentation / erasure based regularization				
Mean teacher	76.97% ± 0.44	38.90% ± 0.21	30.04% ± 0.27	17.81% ± 0.08
RandErase	70.48% ± 0.47	35.61% ± 0.18	28.21% ± 0.07	16.71% ± 0.13
CowOut	68.86% ± 0.35	38.82% ± 0.20	27.54% ± 0.13	16.46% ± 0.10
Mix based regularization				
CutMix	64.11% ± 1.18	30.15% ± 0.26	24.08% ± 0.11	16.54% ± 0.08
CowMix	**57.27% ± 0.60**	**29.25% ± 0.21**	23.61% ± 0.13	**15.73% ± 0.07**

Table 6.4: Results on CIFAR-100 test set, error rates as *mean* ± *std-err* of 5 independent runs.

7 Deep Neural Networks for Analysis of Fisheries Surveillance Video and Automated Monitoring of Fish Discards

In this chapter we describe CatchMonitor; a system developed for automated by-catch (fish discarded on-board fishing trawlers) quantification. It was developed as part of the SMARTFISH project, funded under European Union Horizon 2020.

7.1 Motivation

The quantity of fish discards on board fishing trawlers is currently estimated via measurements obtained during on-board observer sampling. The quantity of discard data is therefore limited by the availability and cost of the observers. In contrast, more precise measurements of the quantity of catch landed at port are available as it is weighed in order to ensure compliance with the trawlers individual quota. Quota is assigned according to the total allowable catch (TAC) quota established by the Common Fisheries Policy of the European Union.

A pilot catch quota management scheme (CQMS) in the UK aimed to improve the quality of discard estimations by installing electronic monitoring systems on-board participating trawlers within the Scottish demersal fishing fleet. These systems included video surveillance cameras monitoring the conveyor belts on which fish are processed or discarded. Marine Scotland Science analysts reviewed the numbers, sizes and species of fish caught per vessel by sampling each vessel's video record when it returned to port [Needle et al., 2014]. Manually counting, measuring and identifying the species of the discarded fish has proved to be laborious and time consuming, motivating the development of a computer vision system designed to analyse the footage automatically.

The intended end result is a system that supports the experts by automating as much of the tedious and expensive manual analysis as possible. We can therefore outline the main requirements of the computer vision component of our prototype system; firstly to detect and count fish leaving the discard chute and secondly to classify a subset of commercial species. Such a system must be robust to the multiple occlusions and unstructured scenes that arise in the unconstrained environment of a commercial fishing trawler; fish are randomly oriented and frequently occlude one another and the view of the working area may be occluded by fishers processing the catch. (see Figure 7.1).

7.2 Overview

CatchMonitor consists of two sub-systems, with the division between them separating the training and inference tasks, as will be familiar to machine learning practitioners. We will discuss the inference sub-system first, as the requirements of its components inform the choices made for the training sub-system.

Inference. The intended output of the inference sub-system is a per-species count of discarded fish; specifically the fish that reach the end of the conveyor belt and fall into the discard chute. This requires the following components:

- *Detection and instance segmentation:* individual fish must be accurately detected and segmented in order to support the later components that must identify their species and track them. Instance segmentation – as opposed to detection – is necessary as fish are randomly oriented and frequently occlude one another in the surveillance footage. This component is discussed in Section 7.5.
- *Species identification:* fish found by the detection and segmentation system must be classified according to species. The development of this classifier and its performance is discussed in Section 7.6.
- *Tracking:* individual fish are tracked during the time in which they are visible in the video. Species class predictions can be generated for each frame, with the class predictions averaged over the time in which an individual is visible, effectively ensembling predictions over multiple frames. Once a fish reaches the end of the belt we consider it to have been discarded, so we add it to the discard count.

Training. Marine Scotland provided us with the surveillance footage that was gathered during their CQMS pilot study [Needle et al., 2014]. From this we needed to create a training set in order to train the models used by the components of the inference system described above. Still images were selected and extracted from the videos for annotation. A web based annotation tool was developed for the purpose of allowing Marine Scotland and CEFAS stuff to create ground truth annotations for instance segmentation and classification. The dataset and the tool are described in detail in Section 7.3.

As discussed in Chapter 2, deep learning based approaches for image classification, object detection and image segmentation have achieved state of the art results. Their impressive performance however comes at the cost of requiring large quantities of annotated training data. With a view to alleviating this burden, we experimented with the semi-supervised learning and domain adaptation approaches discussed in Chapters 4 and 5.

Our surveillance footage comes from several different vessels, each with a distinct visual appearance due to different lighting, camera angles and belt materials (see Figure 7.1). This domain gap (see Section 2.8) presents a challenge as deep neural networks are prone to over-fitting [Krizhevsky et al., 2012] and will often exhibit poor performance on data drawn from a different distribution to that on which they are trained. We also therefore explore the use of domain adaptation to address this.

7.3 Dataset and data acquisition tools

In this section we discuss the process that we developed in order to extract usable image data from the CCTV video that could be annotated, allowing us to train and evaluate the machine learning components of our system.

We will discuss the source video material, the project web application, calibration and segmentation dataset selection and preparation.

7.3.1 Video sources

The surveillance footage was captured in 800p HD resolution and stored in MPEG-4 format. The videos come from 7 sources; 6 commercial fishing vessels and 1 research vessel operated by Marine Scotland. The footage from the commercial vessels captures the real-world working environment and presents challenging conditions, including occlusions by personnel working at the conveyor belt and the view being obscured by spatter on the dome that covers the camera. The footage from the research vessel is similar in terms of content and layout but provides the opportunity to capture tailor-made footage for the purpose of gathering training data.

The videos from the commercial vessels show the conveyor belt. Fish is loaded onto the conveyor by the fishers, often off-screen. One or more fishers standing at the belt pick up and gut the individual fish that they determine are to be landed at port. Once the saleable fish has been removed and processed, the conveyor belt is activated and it carries the remaining guts, detritus and discarded fish over the edge into a discard chute that drops it into the ocean.

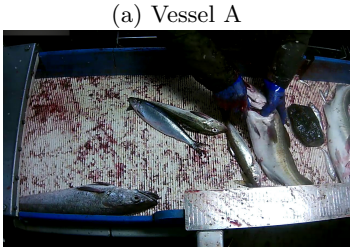
The footage from the commercial vessels consists of the mix of species that was being processed on board the vessel at the time of capture. The footage from the research vessel was specifically produced by Marine Scotland staff by placing large numbers of fish of a known species on the conveyor belt and running it past the camera. Each video from the research vessel contains fish of a single species; this was done for the purpose of training the species classifier, discussed in Section 7.6.4.

The footage is summarised in Table 7.1. Example frames are shown in Figure 7.1.

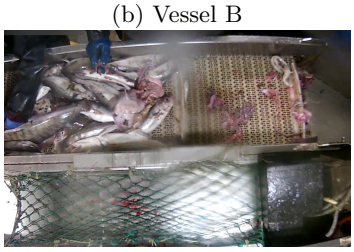
Vessel	Type	Training		Validation		Test	
		# Videos	Time	# Videos	Time	# Videos	Time
Vessel A	Commercial	38	37:30:47	4	4:00:02	5	5:00:02
Vessel B	Commercial	23	22:45:42	5	5:00:02	5	5:00:02
Vessel C	Commercial	26	20:38:26	4	2:58:56	5	3:47:37
Vessel D	Commercial	25	24:14:22	5	4:22:15	5	4:45:50
Vessel E	Commercial	26	2:30:59	9	1:01:42	13	1:01:19
Vessel F	Commercial	8	2:44:13	8	0:35:00	9	0:42:25
Vessel R	Research	54	6:21:40	–	–	–	–

Table 7.1: Summary of video footage. Running times are of the form HH:MM:SS.

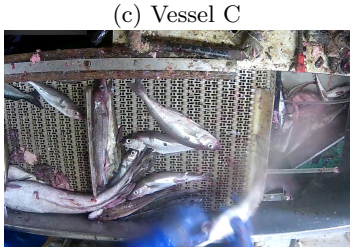
(a) Vessel A



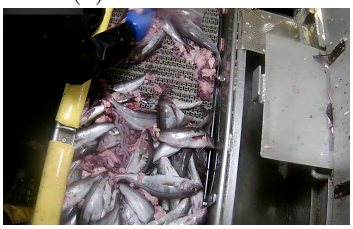
(b) Vessel B



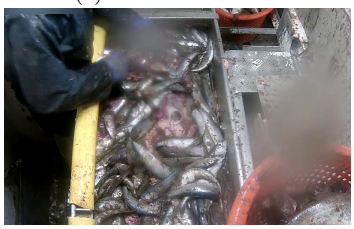
(c) Vessel C



(d) Vessel D view 1



(e) Vessel D view 2



(f) Vessel E

(g) Vessel F

(h) Vessel R

Figure 7.1: Images from each vessel

7.3.2 Web application

To facilitate collaboration between personnel from Marine Scotland, CEFAS and the University of East Anglia, a web application was developed using the Django Framework[1]. The website allows Marine Scotland and CEFAS staff to upload CCTV footage and annotate images for training our computer vision systems (see Section 7.3.3).

7.3.3 Segmentation and species ID training set

A segmentation data set consisting of still frames extracted from the video footage was required in order to train and evaluate the segmentation system. We wished to extract a set of images that cover a representative sample of the fish being processed on the conveyor belt. We wanted to ensure that in between frames chosen for extraction, the motion of the belt carries new as-of-yet unseen material into view. Sampling frames at regular or random intervals is unlikely to suffice as the on-board personnel move the belt according to the amount of fish that they must process; its motion is unpredictable. Furthermore, some videos contain long segments in which the conveyor belt remains empty and does not move. We chose to extract frames such that the belt moves by at least half the length of the visible region of the belt to ensure that the content is sampled as evenly as possible and that the content changes sufficiently between extracted frames. This required a robust estimate of the belt motion, which we discuss in Section 7.4. We should note that there is overlap between successive frames, so some individual fish are visible in more than one training set frame.

Image annotation tool

The images selected for segmentation were uploaded to the web application after which they were manually annotated by Marine Scotland and CEFAS staff. Within this application the labelling tool[2] allows the user to draw polygonal annotations and classify them. The user can select from 39 species of fish, 3 species of crustacean and several non-fish classes such as person, belt structure or guts. There are also classes used to indicate unidentifiable fish or material. The labelling tool can be seen in Figure 7.2.

The annotation tool provides a number of tools to support the user in creating polygonal annotations. In addition to manually drawing polygonal labels one vertex at a time, the user can use a brush tool to quickly cover regions. The user may also choose to add to or remove from a label via the use of Boolean operations, provided by the polybooljs [Connelly, 2017] library.

While the aforementioned tools allow for a fast and effective work-flow, manually annotating fish is a labour intensive task, especially in dense images such as the one seen in Figure 7.2. To this end, we use an implementation[3] of the DEXTR [Maninis et al., 2018] algorithm to significantly speed up the outlining process. The DEXTR tool allows the user to identify a fish by clicking where its outline touches the four edges of a bounding box surrounding

[1] Available from `https://djangoproject.com`
[2] Available from `https://github.com/Britefury/django-labeller`
[3] Available from `https://github.com/Britefury/dextr`

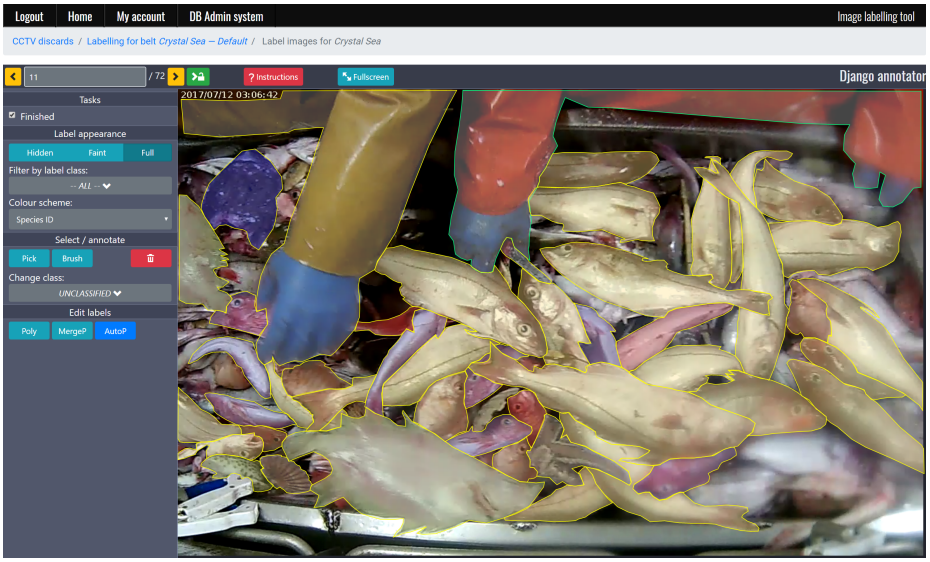


Figure 7.2: Web-based segmentation annotation tool

it. The request is passed from the client-side browser application to the web server, that passes it on to a separate machine that has a GPU that is used to execute the DEXTR model. The output of the DEXTR model is a mask image that is vectorized and returned via the web server to the client-side browser application. This typically happens in under a second. Manual outlines created using the annotation tool are used to fine-tune a DEXTR model trained on the Pascal VOC 2012 [Everingham et al., 2012] data set, resulting in a model that produces high quality outlines for fish.

The manual annotation process was further automated by using a Mask R-CNN model trained using existing annotated data to predict annotations for un-annotated images. Once between 100 and 200 images had been manually annotated for each belt, we found that a segmentation model trained using these annotations was able to automatically annotate the majority of fish to a satisfactory standard. We generated automatic annotations for as-of-yet un-annotated images and placed them on the website to serve as a starting point for the annotators. This saved considerable effort as the annotators only needed to annotate the fish that had been missed by the model or fix its mistakes. The improved annotations could then be added to the training set that was used to train a new and more accurate segmentation model and a more accurate DEXTR model, resulting in a cyclic process.

Data

The training data for the segmentation system consists of 3,425 annotated frames drawn from videos from the 7 vessels and is summarised in Table 7.2. While many more frames were extracted from the videos, this is the subset that has been annotated so far. The rest remain as of yet un-annotated due to the labour involved.

We note that our segmentation dataset, consisting of ∼3,400 training images is somewhat smaller than the well known CoCo 2017 [Lin et al., 2014] dataset, consisting of ∼123,000

Vessel	# annotated images	# annotated fish
Vessel A	659	3779
Vessel B	1057	4843
Vessel C	559	5007
Vessel D	463	8058
Vessel E	279	3103
Vessel F	208	10440
Vessel R	200	2444
Total	3425	37674

Table 7.2: Segmentation training set

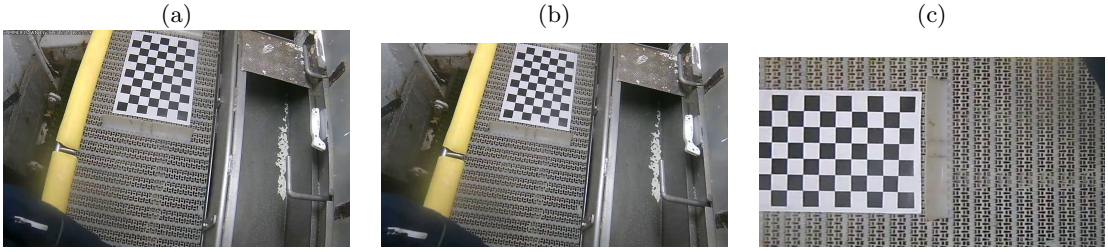


Figure 7.3: The belt extraction and calibration process: (a) checker board on belt, (b) with lens distortion removed and (c) with perspective warp used to transform belt into rectilinear space and exterior cropped out

images. We attribute the fact that we are able to obtain adequate performance from a dataset of this size to the fact that our images are less diverse and that we are detecting objects of a single class, as opposed to the 81 classes present in CoCo 2017.

7.4 Calibration and belt motion estimation

An estimate of the belt motion was required for selecting frames for the training set (Section 7.3.3) and for object tracking and discard quantification (Sections 7.7 and 7.8). Our approach for motion estimation is based on matching key-points in consecutive frames and estimating the transformation between them. We found that the accuracy of this approach can be improved by simplifying the transformation that must be estimated. We first removed lens distortion from the image (see Section 7.4.1). Subsequently we applied a perspective warp and crop that extracts the region of the image that covers the conveyor belt and transforms it so that it is in rectilinear space. Our transformation was chosen such that the extracted belt moves horizontally to the left when active. Thus, the transformation that we wish to estimate is reduced to being only a translation. The lens distortion removal and perspective warp and crop are illustrated in Figure 7.3.

7.4.1 Lens distortion correction

The surveillance cameras on-board fishing vessels often use fish-eye lenses to increase their field of view. This introduces a curved distortion to the image that complicates later stages

of the system. The OpenCV library [Bradski, 2000] provides functionality for automatically estimating lens distortion parameters and removing it from images.

The lens distortion estimation algorithm within OpenCV requires that a printed checker board pattern is captured at various positions within the cameras field of view. The cell corners are detected and their positions are used to estimate the lens distortion. We provided Marine Scotland and CEFAS staff with a checker board pattern and a procedure for capturing calibration footage on-board fishing vessels.

The CCTV systems on-board fishing vessels are configured for capturing video rather than still images. Localising the checker board in all frames in which it is visible in a calibration video typically results in several hundred detections. The lens distortion estimation algorithm run-time scales in a super-linear fashion with respect to the number of detections used, failing to complete within a reasonable time when presented with large numbers of detections. In order to avoid having to select frames manually we developed an algorithm that selects detections that are dissimilar from one another.

We divide the image into 50×50 pixel cells and quantize the co-ordinates of the checker board corners, generating a map that specifies which cells are covered (`histogram2D` in the algorithm). If more than 22% (determined by trial and error) of the cells covered by this checker board have not been covered by a previously selected checker board detection we add it to our selection. The algorithm is given below:

Algorithm 4 Lens estimation detection selection algorithm

$covered \leftarrow \text{BOOLEANARR2D}(num\_cells\_y, num\_cells\_x)$
$selected\_dets \leftarrow []$
for each $det \leftarrow detections$ **do**
  $det\_coverage \leftarrow \text{HISTOGRAM2D}(det, cell\_size)$
  if $\text{MEAN}(det\_coverage \wedge \neg covered) \geq 22\%$ **then**
    $covered \leftarrow covered \vee det\_coverage$
    $selected\_dets.\text{APPEND}(det)$
  end if
end for

7.4.2 Belt warp and calibration

The perspective warp used to transform the belt into rectilinear space – as seen in Figure 7.3(a) – was estimated using the checker board used for lens distortion estimation. It was laid flat upon the belt, after which the checker board localization algorithm within OpenCV was used to find it. The co-ordinates of its cell corners were used to estimate the perspective transformation required to transform the checker board into a fixed rectangular shape of a known size. The checker board was printed on A3 paper, giving it known physical dimensions, resulting in a transformation that removes the perspective distortion and scales the image of the belt to a known physical distance to image space ratio. A tool was developed within Jupyter Notebook [Kluyver et al., 2016] that allows the user to correct for any misalignment and crop out the region corresponding to the belt.

7.4.3 Belt motion estimation

The process of panorama creation – discussed in Section 2.11 – forms the basis of our approach for belt motion estimation. The panorama creation process matches key points between partially overlapping images in order to compute a homography to align the images prior to compositing. We can re-purpose this approach for motion estimation by matching key points on the conveyor belt between subsequent frames and estimating the inter-frame motion.

The accuracy of the estimate of the fundamental matrix used to estimate motion can be improved by constraining the model and reducing the number of parameters as much as possible. By applying lens distortion removal and belt extraction (Section 7.4.2) prior to motion estimation, the belt motion is constrained to a horizontal translation. We found that using a two-dimensional translation model instead of a one-dimensional horizontal translation resulted in better estimates. The vertical component was necessary, most likely to account for slight inaccuracies in the belt extraction transformation. After estimating the translation, its horizontal component x was clamped to lie in the range $[-x_{max}, 0]$, where x_{max} is the maximum per-frame horizontal displacement, computed as a product of the maximum belt speed of $0.9ms^{-1}$, the image space to physical distance ratio and t the reciprocal of the frame rate. The y component was clamped to lie in the range $[-y_{max}, y_{max}]$, where the maximum vertical displacement y_{max} is the maximum of a vertical tolerance value of 3 pixels or $x \tan \theta_{max}$, where θ_{max} is the angular tolerance of 20°.

We used the implementation of ORB [Rublee et al., 2011] provided by OpenCV [Bradski, 2000] for key point detection and feature extraction. The Scikit-Image [van der Walt et al., 2014] implementation of RANSAC [Fischler and Bolles, 1981] was preferred as its API allows the model being estimated to be customized, allowing us to use the constrained translation model described above.

We tested key-point and feature extraction on images both with and without the lens distortion correction and belt extraction processes described above. Performing the key point and feature extraction step without belt extraction replaces a fairly costly image warping step with the comparatively cheap operation of applying the lens distortion correction and belt extraction transformations to 500 key point co-ordinates. We found however that doing this could result in the key point detection algorithm focusing on irrelevant parts of the image, such as the outline of gloves worn by fishers set against the conveyor belt. Given that the motion model is estimated using the key points, this would bias the estimate towards irrelevant parts of the image, resulting in a poor estimate. Detecting key points on images subsequent to belt extraction however avoided this problem, so this was our preferred method. This is illustrated in Figure 7.4, where a considerable number of key-points are placed on the outline of the hand in *camera space*, causing the motion estimate to track the motion of the hand rather than that of the conveyor belt. Less key-points are placed on the hand when using *rectified belt space*, causing the motion estimate to track the belt.

(a) (b)

Figure 7.4: Contrast between key-point detection on (a) camera-space images vs. (b) belt-space images. Green circles are key-points matched to key-points in the previous frame, red circles are key-points that were not successfully matched.

7.5 Instance segmentation

An effective instance segmentation algorithm is a pre-requisite to the successful operation of the complete system as later stages rely on accurate detection and segmentation in order to reliably classify the species of individual fish.

During the course of the project we experimented with a variety of approaches to solving this problem. Our first attempt [French et al., 2015] combined semantic segmentation [Long et al., 2015a] with contour detection. The semantic segmentation model identified regions of the image containing fish, with the caveat that multiple fish that touch or overlap – as occurs frequently in our CCTV footage – will form a contiguous region. A contour detection model [Ganin and Lempitsky, 2014] located edges of fish, guiding the Watershed algorithm [Beucher and Meyer, 1993] in order to split contiguous regions, separating individuals from one another. As stated in Section 2.5.3 this approach is unreliable as small gaps in detected contours can prevent instance separation, as the Watershed algorithm is a flood-fill based approach. This problem, was exacerbated by the domain gap that exists between footage obtained from different conveyor belts due to differences in appearance and lighting. As a consequence our prior approach required us to train separate segmentation models for each conveyor belt and use carefully tuned post-processing to mitigate this problem.

Mask R-CNN [He et al., 2017] proved to be an effective and efficient instance segmentation algorithm, hence we adopted it for use in our system[4]. As stated in Section 2.5.3 it combines object detection with mask prediction and is therefore much more robust than our prior *segment and separate* approach. It generates high quality labels as seen in Figure 7.5. Furthermore in contrast to our prior work [French et al., 2015], we are able to use a single segmentation model for all vessels.

[4]We use the COCO [Lin et al., 2014] pre-trained implementation of Mask R-CNN provided by the *torchvision* library that is developed by the PyTorch [Chintala et al., 2017] team. It produces good results and trains quickly.

Figure 7.5: Instance segmentation applied to a frame from footage from Vessel R (research vessel); the blue labels are generated automatically

As stated in Section 7.3.3 the segmentation system was used to automatically annotate images on the labelling tool section of the project web application, after which mistakes in the annotations could be fixed manually. We maximised the quality of the automatically generated annotations using test-time augmentation [He et al., 2017]; each image was segmented 8 times, with random augmentation consisting of horizontal and vertical flips, lightening and darkening, scaling and rotation. The resulting predictions were averaged, increasing their accuracy. Doing so comes at significant computational cost, so this is only feasible for off-line use when the improved accuracy is worth the additional run-time.

7.5.1 Separate species identification

The object detection network that forms the basis of Mask R-CNN [He et al., 2017] incorporates a classifier that predicts the class of detected objects and a multi-class mask head that learns class specific shapes for segmentation. In principle, this could be used to perform fish detection, segmentation and species identification in a single pass. We however opt to use separate networks, using a single class Mask R-CNN network for fish detection and segmentation only, followed by a classifier for species identification. We do this for two reasons that we will now explain.

Identifying the species of fish in our surveillance footage requires annotators with the relevant training and experience. In contrast outlining individual fish for segmentation can be performed by a wide variety of individuals. To support this we allow annotators to outline fish in an image without specifying their species. As a consequence many images in our dataset have fish outlined for segmentation but with some individuals having no assigned species. Training a multi-class Mask R-CNN model requires per-object class labels in order to select the class-specific bounding box regressor and mask head to optimise for each object. As a consequence images with partial species annotation would not be usable for training a multi-class Mask R-CNN model.

Furthermore, as we will state in Section 7.6.1 we were able to improve the performance of our classifier by rotating the images of segmented fish so that they lie horizontally, as doing so eliminates a source of irrelevant variation. Mask R-CNN does not provide a mechanism for altering the orientation of objects prior to classification.

For these reasons we train our Mask R-CNN model to detect and segment objects of a single fish class and identify species in a subsequent step (see Section 7.6).

7.5.2 Training procedure

While training our segmentation network we apply stochastic augmentation, consisting of random horizontal and vertical flips, random rotations between -45°and 45°, applying a random uniform scale factor in the range of 0.8 to 1.25 and randomly re-colour the image using the `ColorJitter` augmentation provided by the *torchvision* library.

We split our dataset into 90% for training and 10% for validation. We train for 150,000 iterations. We report the mean average precision (mAP; Lin et al. [2014]) score for the validation samples in our logs. We use a learning rate of 10^{-4} for the new randomly initialised later layers and 10^{-5} for the pre-trained layers that come from the *torchvision* [Chintala et al., 2017] Mask R-CNN implementation. We randomly crop 512×512 pixel regions from our rectilinear belt images and build mini-batches of crops from 4 randomly chosen images during training. We train our models on a single nVidia GeForce 1080-Ti GPU.

In addition to the bounding box non-maximal suppression used in Mask R-CNN [He et al., 2017] we apply *mask NMS* to the masks predicted during inference. If the proportion of the pixels predicted as belonging to an object are already occupied by other objects with a higher predicted confidence is greater than the mask NMS threshold τ_n, the lower scoring object is ignored. While we found that a detection confidence threshold τ_d of 0.85 maximized performance for inference on still images, a different value yielded an optimal result for discard counting (see Section 7.8.5). We use a lower threshold of 0.5 for generating labels for the annotation website, decreasing the number of false negatives at the expense of an increase in false positives. Requiring annotators to select false positive detections and mark them as *not fish* requires significantly less manual effort than drawing missing annotations resulting from false negatives. On balance the lower threshold reduces manual annotation effort in spite of the decreased mean average precision (mAP).

7.6 Species identification

In this section we describe our species classifier, the development of the dataset required for training and our evaluation of its performance.

7.6.1 Classifier

Our species classifier is a 50-layer residual network [He et al., 2016] adapted and fine tuned using transfer learning. It operates on images of individual fish that are identified by the instance segmentation system (see Section 7.5).

Our annotators requested that our species list should include a number of *generic* classes for instances in which the precise species of an individual fish cannot be identified due to the viewing angle or occlusion. For example, the 'generic gurnard' class is used when the annotator cannot determine if the fish is a red gurnard or a grey gurnard. During training

Species	Vessel A	Vessel B	Vessel C	Vessel D	Vessel E	Vessel F	Total
Cod	123	146	99	73	0	6	447
Haddock	308	459	713	1855	51	2112	5498
Whiting	43	145	95	126	20	17	446
Saithe	1002	11	368	96	0	0	1477
Hake	283	32	198	48	3	14	578
Monk	5	15	8	3	224	79	334
Mackerel	16	0	15	1	0	0	32
Horse mackerel	33	0	48	19	0	0	100
Norway pout	102	44	24	60	6	3	239
Plaice	4	142	18	1	361	8	534
Long rough dab	5	3	0	1	25	0	34
Common dab	0	10	7	1	28	0	46
Grey gurnard	146	18	29	36	58	443	730
Red gurnard	0	1	0	0	52	156	209
Gurnard (generic)	237	763	311	269	624	2897	5101
Dover sole	0	0	0	0	355	5	360
Lemon sole	3	19	0	2	131	7	162
Dog fish	1	0	0	1	148	226	376
Sea bass	0	0	0	0	1	0	1
John Dory	0	0	0	0	0	15	15
Megrim	0	2	0	0	8	98	108
Ling	12	2	3	1	0	0	18
Herring	23	1	193	192	1	0	410
Bib	0	0	0	0	411	17	428
Brill	0	0	0	0	14	8	22
Turbot	0	0	1	0	0	2	3
Boar fish	0	0	0	0	4	998	1002
Argentines	26	67	31	3	0	0	127
Witch	0	1	0	0	0	1	2
Catfish	0	1	1	0	0	0	2
Cuttlefish	0	0	0	0	3	1	4
Norway haddock	4	0	0	0	0	0	4
Red mullet	0	0	0	0	1	3	4
Conger eel	0	0	0	0	0	1	1
Common dragonet	0	0	0	0	138	65	203
Skate/ray	51	273	18	28	18	43	431
Squid	0	0	0	0	0	74	74
Shark	2	0	0	17	9	2	30
Fish unidentifiable	763	894	838	2257	200	2815	7767
Flat (generic)	28	163	21	68	130	128	538
Total	3220	3212	3039	5158	3024	10244	27897

Table 7.3: Summary of species identification dataset from commercial vessels. Note that unsupervised samples are not included.

we convert ground truth classifications into one-hot vectors that are used to compute cross-entropy loss. A *generic* class that covers c classes is represented as a c-hot vector, in which a value of $\frac{1}{c}$ is assigned to each class covered by the generic class.

The list of species that could be chosen by the annotators grew throughout the lifetime of the project, with new species being added as the marine biologists at Marine Scotland Science and CEFAS requested them. At the time of writing, our system supported 42

	Cod	Haddock	Whiting	Saithe	Hake	Mackerel	Horse mackerel
# of fish	264	2687	3173	132	65	450	96
# of videos	3	19	14	3	2	1	1

	Norway pout	Plaice	Long rough dab	Common dab	Grey gurnard	Red gurnard
# of fish	1379	505	307	310	287	10
# of videos	2	3	1	2	3	1

Table 7.4: Summary of species identification dataset from the research vessel

species that were requested by Marine Scotland and CEFAS staff. Three of them – namely brown crab, European lobster and crawfish – are not listed in Table 7.3 as no examples of them were found by our annotators.

We found careful pre-processing of images of individual fish to be essential for good classification performance. While the fish in our surveillance footage are arbitrarily oriented, we found that rotating images of individual fish so that they lie horizontally eliminated a source of irrelevant variation, improving accuracy. We used the `regionprops` function from the Scikit-Image [van der Walt et al., 2014] library to estimate the orientation from the shape/mask predicted for each fish and rotate it so that the longest axis lies horizontally. This ensures that most fish lie horizontally, although they vary in horizontal and vertical direction (left-to-right or right-to-left, upside-down).

Individual fish of a given species can vary considerably in size. We scale images of fish to a fixed size to remove scale as a source of variation, therefore relieving the classifier of the need to learn to recognise useful visual features at a variety of scales. While doing this discards size information – which could be helpful for discriminating between some species – we found that re-introducing it by providing the size to the classifier did not improve performance.

Each fish was scaled so that it fits within a fixed image size, leaving a border whose width is $1/16$ of the image size. For example, for an image size of 224×224 pixels, the fish is scaled so that it fits within a 196×196 region that is centred within the image. The masks predicted by the segmentation system are often imperfect and may exclude parts of a fish that are helpful for classification. To counter this, we found that expanding the mask in all directions by 5% of the region size (e.g. 5% of 196 pixels = 10 pixels) using binary dilation improved performance. The border therefore leaves space for the mask dilation and for up-scaling transformations applied during data augmentation. After extracting, rotating and scaling the relevant region from the source image, we modulated the image with the mask, setting pixels outside of it to 0, removing any distracting cues from parts of the image outside the bounds of the fish. We note that the mask was applied after standardisation (zero-mean and unit-variance, using the mean and standard deviation values used with the pre-trained ImageNet classifiers provided by PyTorch [Chintala et al., 2017]).

7.6.2 Training

We trained our classifier using the Adam [Kingma and Ba, 2015] optimization algorithm with a learning rate of 1×10^{-4}, with the net layers carried over from the ImageNet pre-trained network using a learning rate of 1×10^{-5}. We use over-sampling to compensate for class imbalance, by drawing samples with a frequency that is inversely proportional to the frequency of the ground truth class. During training we apply stochastic data augmentation consisting of rotations in the range $[-10°, 10°]$, uniform scaling in the range $[0.8, 1.2]$, random translations with a magnitude of 16 pixels and random horizontal and vertical flips. This augmentation scheme was chosen manually to mimic the variance in orientation and size estimation above caused by varying mask quality.

7.6.3 Semi-supervised learning

The fact that only a subset of our available data has been annotated presents a use case for semi-supervised learning typical of many practical applications. Our prior experience with consistency regularization discussed in Chapters 4 and 5, combined with state-of-the-art results reported in the literature [Sohn et al., 2020; Xie et al., 2019] and its simplicity suggest it as a good approach. Following our work in Chapters 4 and 5, we adopt the Mean Teacher algorithm [Tarvainen and Valpola, 2017] combined with confidence thresholding (see Sections 4.1.2 and 5.6.1). We evaluate two sources of perturbation; CowMask [French. et al., 2022] and RandAugment [Cubuk et al., 2020].

When using RandAugment, we follow the approach of Sohn et al. [2020], using two augmentation schemes; weak and strong. Our weak augmentation scheme uses the scheme described above in Section 7.6.2 above and is applied to samples that are passed to the teacher network for generating pseudo-targets. We use RandAugment for strong augmentation that is applied to samples passed to the student network.

We use CowMix, in which CowMask is used to mix two samples in a similar fashion to that used for semi-supervised semantic segmentation as described in Chapter 5. The CowMix approach is described in French. et al. [2022]. We shall now briefly describe the approach. A batch of samples is augmented using our standard augmentation scheme are passed through the teacher network to produce pseudo-targets. The samples are paired in a cyclic fashion (0 with 1, 1 with 2, ..., $n-2$ with $n-1$, $n-1$ with 0) and mixed using randomly generated CowMasks. For each mixed image, the proportion that came from the first and second input is used to mix the corresponding probability vectors predicted using the teacher network. The mixed images are passed through the student network and consistency loss encourages its predictions to match the mixed teacher predictions.

7.6.4 Training data

Our species identification training data is drawn from footage from the commercial vessels and from the research vessel.

A summary of the species identification training data broken down by vessel and species is given in Tables 7.3 and 7.4. We note the discrepancy between the number of fish whose species has been manually identified in these datasets and the segmentation dataset outlined

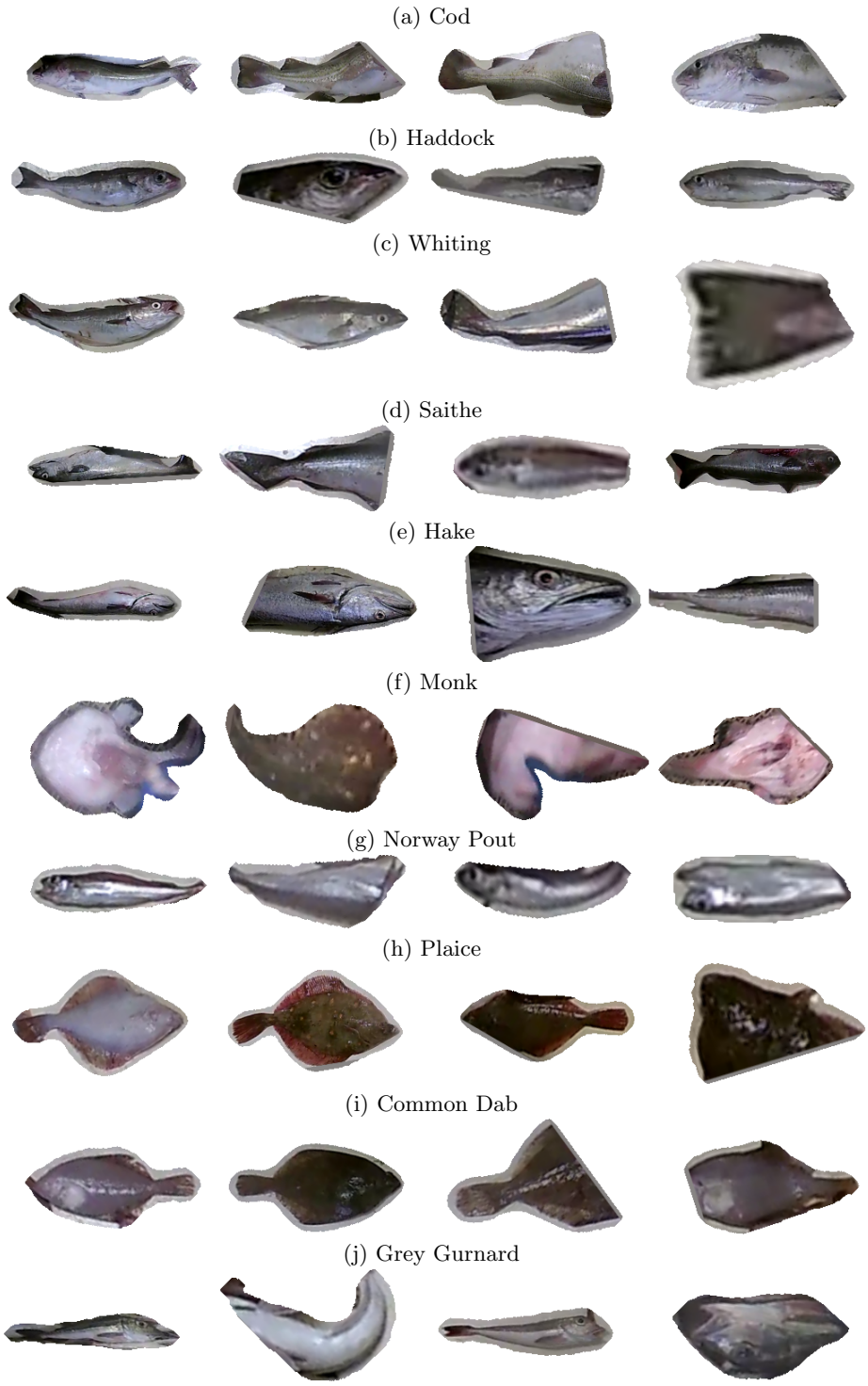


Figure 7.6: Examples from the species identification dataset. All fish were from the single species research vessel footage, apart from monk which were taken from commercial footage. Samples were chosen to illustrate that the classifier often receives only a partial fish or one whose orientation hides useful details.

in Table 7.2. This is due to many of the fish in the segmentation dataset not having a species assigned to them.

The commercial training samples were drawn from commercial footage and their species was determined manually. This is a time consuming and laborious process.With a view to addressing this, Marine Scotland staff placed large quantities of fish of known species on the research vessel conveyor belt and ran it past the surveillance camera. Applying the segmentation system allowed us to extract large numbers of training images of a known species class, resulting in the research training samples summarised in Table 7.4. This further illustrates the advantage of separating segmentation and species classification into separate steps, as mentioned in Section 7.5.1.

The time consuming nature of the manual annotation process is easy to see by comparing Tables 7.3 and 7.4. After two years of manual annotation, the data extracted from footage from commercial vessels contains approximately 22,000 fish (Table 7.3) with manually identified species. In contrast, the ∼9,000 fish (Table 7.4) that comprise our research vessel dataset were extracted from videos captured during a single fishing trip and required the manual outlining of the fish in only 200 images.

A number of manually annotated images within the commercial dataset have segmentation annotations but incomplete or no species annotation. As a consequence there are labels that are known to be fish of an as-of-yet unknown species. We use these labels to generate unsupervised samples used for semi-supervised learning. Furthermore a large number of images extracted from videos from commercial vessels do not have manual segmentation annotation. For these images we apply the segmentation segmentation system to generate automatic segmentation labels. Given that no species identification has been done for them, they too are used to generate unsupervised samples.

It should be noted that the complex and unstructured scenes in our CCTV footage frequently feature fish that are oriented such that useful discriminative features or parts are hidden from view or fish that are only partially visible due to being occluded by overlapping fish or personnel working at the belt. Operating in these challenging conditions is one of the challenges posed by this project. Selected examples from each species are shown in Figure 7.6.

7.6.5 Performance evaluation

To understand the performance of our classifier we evaluate it in three scenarios. Firstly, we train and test on commercial samples, using 4-fold cross validation. We split the samples from each vessel into four folds (see *Train and test on commercial samples* below for our precise method), therefore using samples from all vessels for training and testing. We expect this to be a challenging benchmark given the significant class imbalance (see Table 7.3) and the challenging nature of the input images (see Figure 7.6).

Secondly, we train and test on commercial samples, using leave-one-vessel-out cross validation; the samples from one vessel are held out for testing while those from the remaining vessels are used for training. This scenario is more representative of a system deployed in the field that must operate on samples from a new vessel for which no ground truth

annotations are available. We expect this to be a somewhat more challenging problem due to the domain gap resulting from the different lighting conditions and appearance of the footage between the training vessels and the test vessel.

In our final scenario we train on samples drawn from footage from the research vessel test on samples from the commercial vessels. This is by far the most challenging scenario for the classifier as it will be trained on samples that exhibit relatively little variation in lighting conditions due to coming from a single source. We hypothesize that there will be a significant domain gap between the research and commercial vessels. This scenario is of particular interest as it is ideal from the perspective of preparing training data due to the reduced annotation effort.

Our dataset exhibits severe class imbalance (see Table 7.3) and the samples from the various classes are not equally distributed between vessels or videos. This results in folds or experiments in which some classes have samples in the test set but none in the training set, or vice versa. We therefore do not consider the performance on classes for which there are no samples in the training set.

A confusion matrix is generated for each train/test split within an experiment. The rows corresponding to classes that are not represented in the training set are zeroed out, to prevent the classifier from being unfairly penalized for failing to identify classes that it was not trained to recognize. The confusion matrices from the cross validation splits within the experiment are summed, after which class accuracy scores are computed.

Train and test on commercial samples

As stated in Section 7.3.3, our approach for selecting video frames to be used as training images will result in some overlap. As a consequence some individual fishes appearing in multiple training frames. These fishes would therefore be depicted in multiple samples in our species ID dataset. Splitting by individual samples presents the possibility of some individual fish being present in both the training and test sets in some folds, albeit from different points of view. We eliminate this possibility by splitting the videos from each vessel into four folds, with the species identification samples being assigned to the same fold as the video from which they came from.

The accuracies obtained when training and testing on commercial samples, using 4-fold cross-validation as stated above are shown in Table 7.5. We provide a full confusion matrix for supervised learning in Figure 7.7. Using semi-supervised learning with RandAugment increases the mean class accuracy from 44.4% to 53.8%, with the full confusion matrix shown in Figure 7.8.

Training approach	Mean class accuracy (%)
Supervised	44.4%
Semi-supervised, CowMix	52.7%
Semi-supervised, RandAugment	53.8%

Table 7.5: Performance of species classification when training and testing on commercial samples.

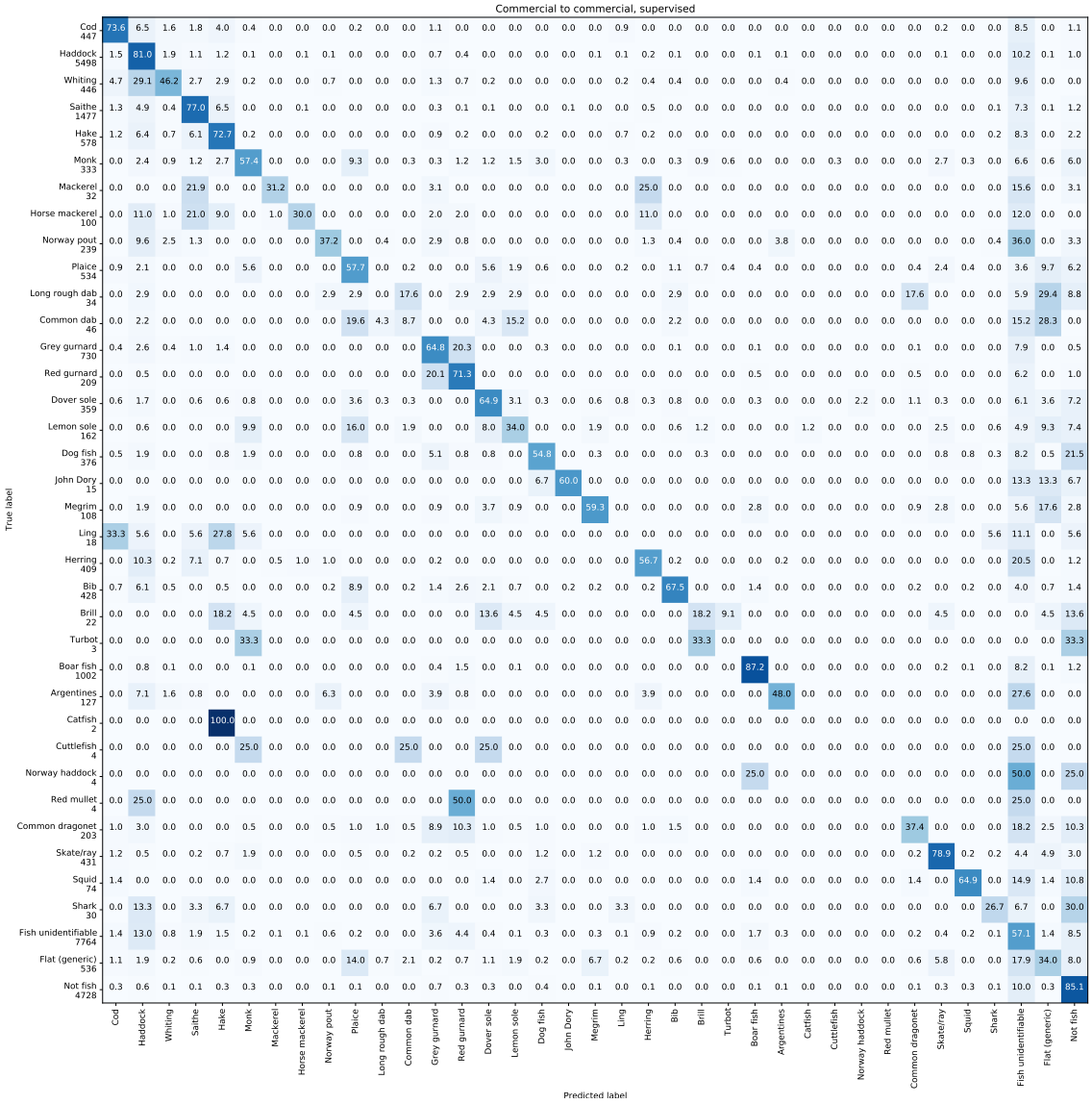


Figure 7.7: Confusion matrix for train and test on commercial samples, 4-fold cross-validation, with supervised training. Mean class accuracy is 44.4%. Values along the left side below each class name indicate the number of samples in that class.

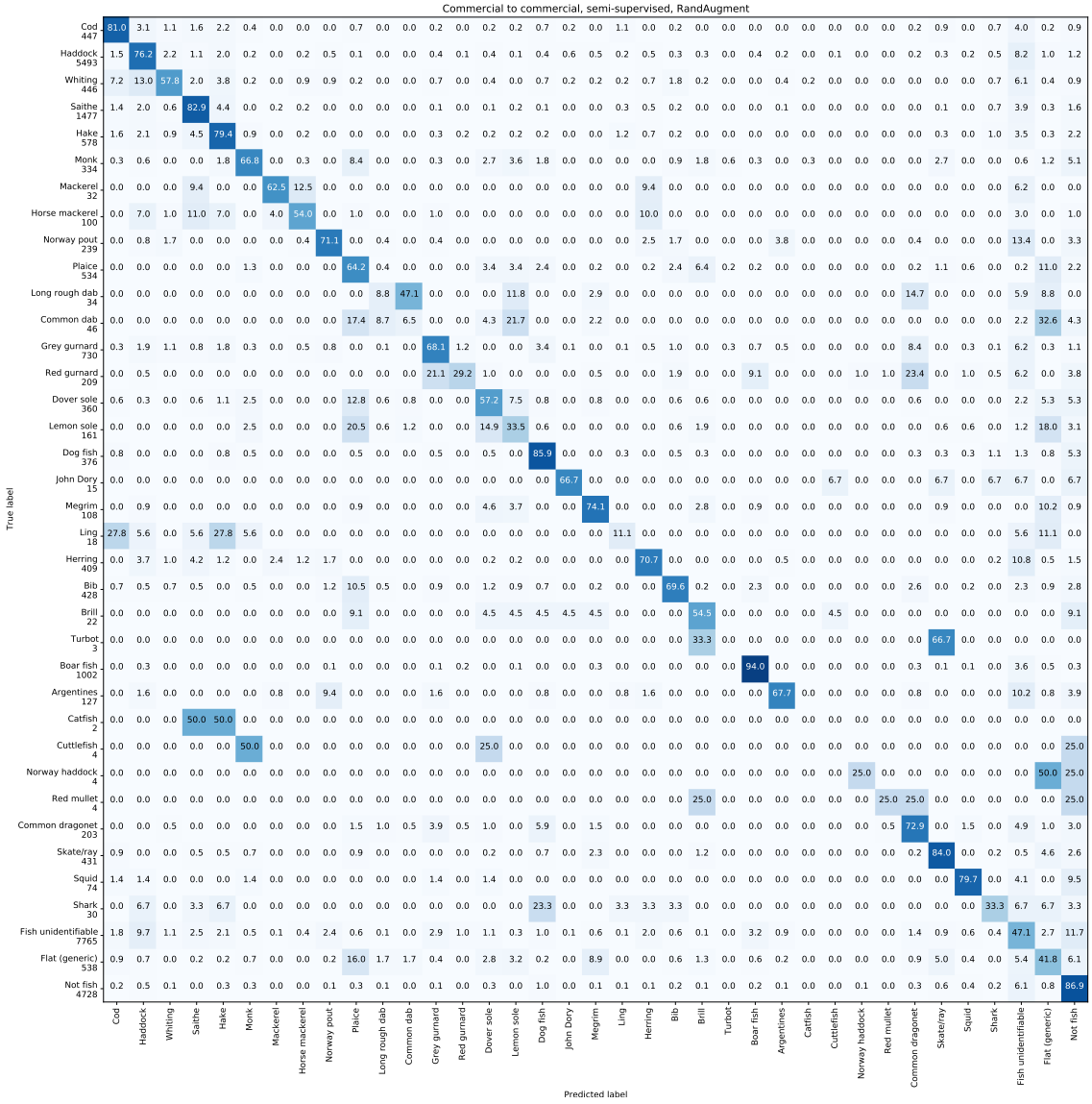


Figure 7.8: Confusion matrix for train and test on commercial samples, 4-fold cross-validation, with semi-supervised learning using Mean Teacher with RandAugment providing perturbation. Mean class accuracy is 53.8%. Values along the left side below each class name indicate the number of samples in that class.

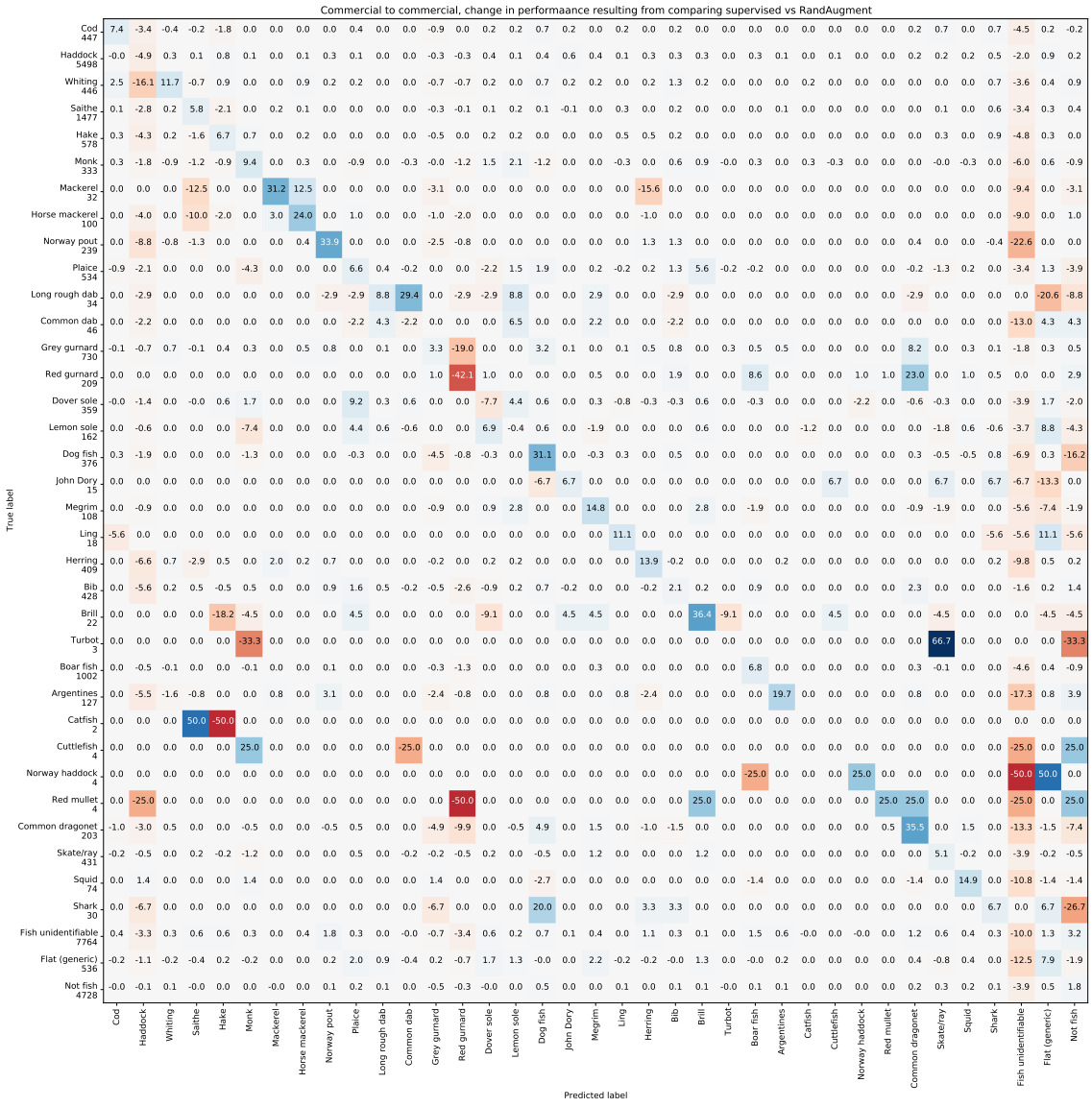


Figure 7.9: Confusion matrix comparison of supervised and semi-supervised learning (with RandAugment) results shown in Figure 7.7 and Figure 7.8. The plot represents the increase (blue) or decrease (red) in predictions resulting from applying semi-supervised learning. The values are the result of subtracting the supervised confusion matrix from the semi-supervised confusion matrix. Values along the left side below each class name indicate the number of samples in that class.

The use of semi-supervised learning results in an 9.4% increase in mean class accuracy score. We illustrate the difference in more detail in the confusion matrix comparison plot shown in Figure 7.9. We observe an improvement in performance indicated by the positive values (an increase in prediction) running along the diagonal and the negative values (a decrease in prediction) outside the diagonal. We note that off-diagonal positive values (resulting in a decrease in overall performance) tend to occur for classes with poor representation, e.g. long rough dab (34 samples), common dab (46), lemon sole(162), turbot (3), catfish (2), cuttlefish (4) and red mullet (4). We note that the use of RandAugment increases the number of the relatively well represented grey (730) and red (209) gurnards mis-predicted as common dragonet (203).

Leave-one-vessel-out cross validation

In practice a system such as the one discussed here would need to exhibit strong performance on footage from vessels for which there is no annotated training data. To assess the potential performance in such a situation we trained six classifiers, each one on samples from five out of six commercial vessels, with samples from the remaining vessel held out for testing. This presents additional challenges due to the domain gap resulting from the differing appearances between the vessels and the difference in class distribution present between vessels. As seen in Table 7.3, many species are present in footage from only two or three of the vessels and often with the majority of instances appearing in footage from one vessel.

The mean class accuracies obtained using leave-one-vessel-out cross validation are presented in Table 7.6. Confusion matrices for supervised learning and semi-supervised learning results with RandAugment are shown in Figures 7.10 and 7.11 respectively. Our results confirm the problems caused by the imbalance in the per-vessel class distribution, with poor class accuracies often occurring for classes with an imbalanced distribution between vessels. The confusion matrix comparison seen in Figure 7.12 resulting from the application of semi-supervised learning shows a performance improvement, demonstrating the effectiveness of semi-supervised learning.

Training approach	Mean class accuracy (%)
Supervised	23.6%
Semi-supervised, CowMix	25.3%
Semi-supervised, RandAugment	30.8%

Table 7.6: Performance of species classification when training and testing on commercial samples, exploring the inter-vessel domain gap with with leave-one-vessel-out cross-validation.

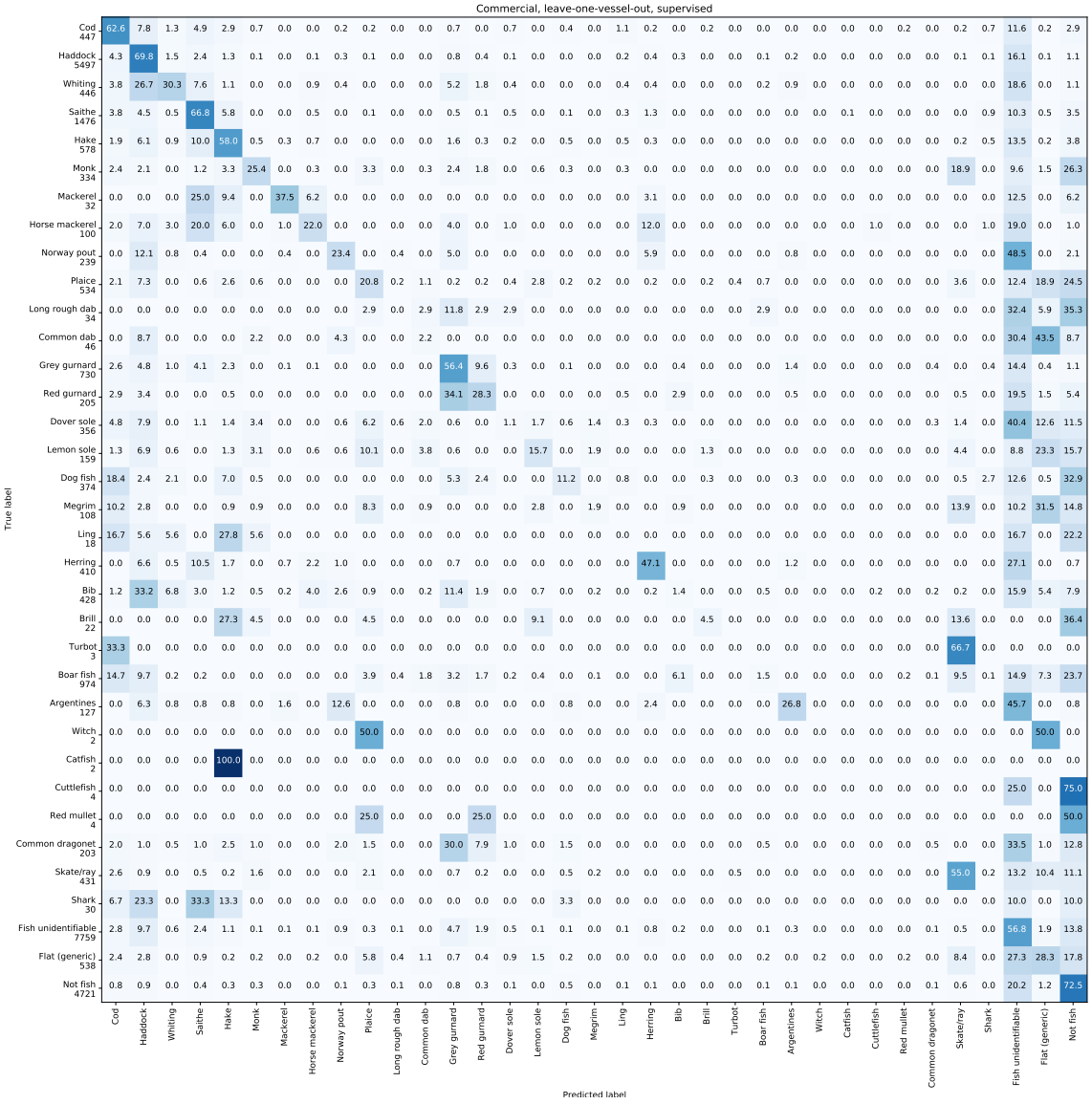


Figure 7.10: Confusion matrix for train and test on commercial samples, exploring the inter-vessel domain gap with leave-one-vessel-out cross-validation, with supervised training. Mean class accuracy is 23.6%.

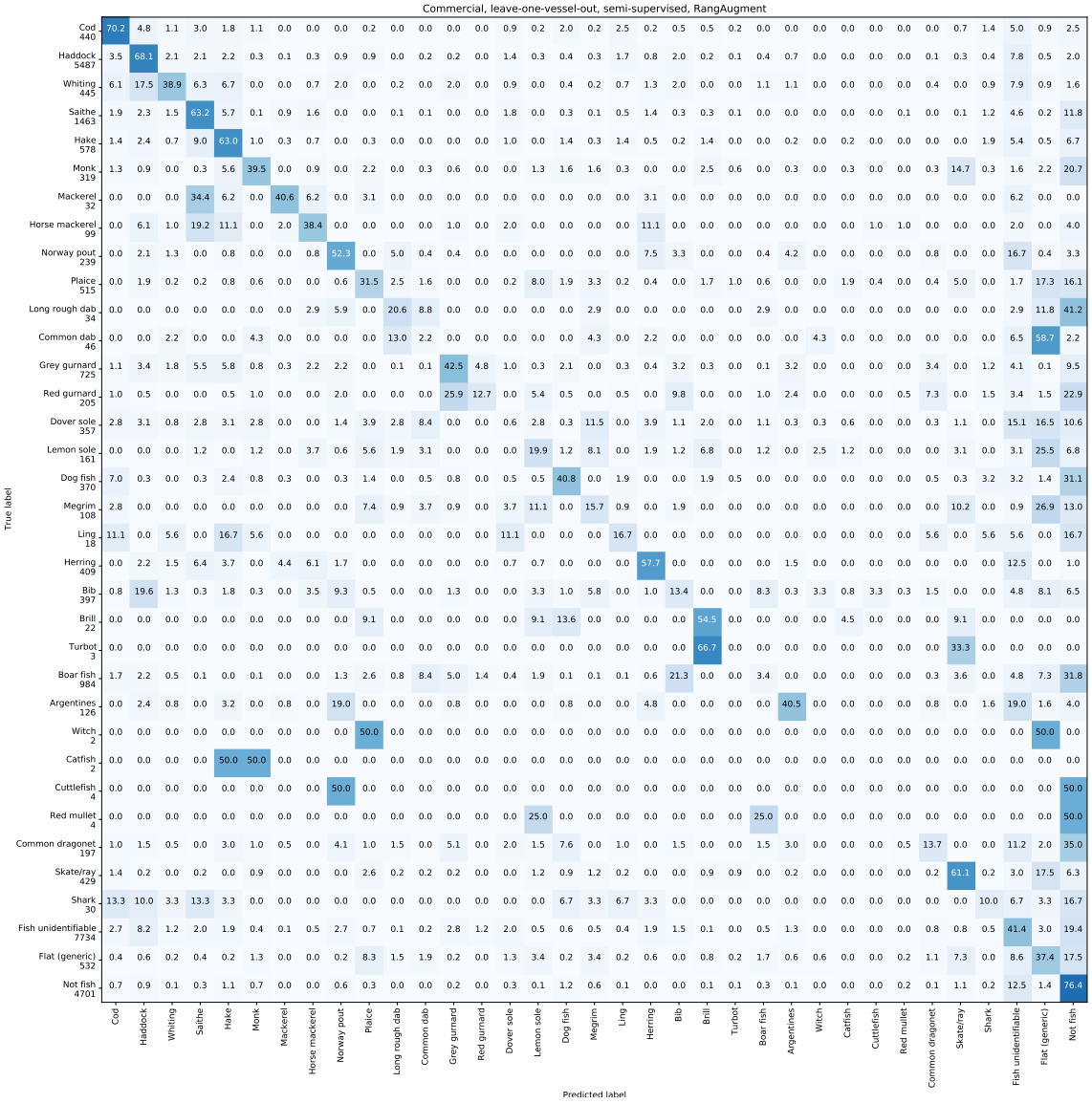


Figure 7.11: Confusion matrix for train and test on commercial samples, exploring the inter-vessel domain gap with leave-one-vessel-out cross-validation, with semi-supervised training and RangAugment. Mean class accuracy is 30.8%.

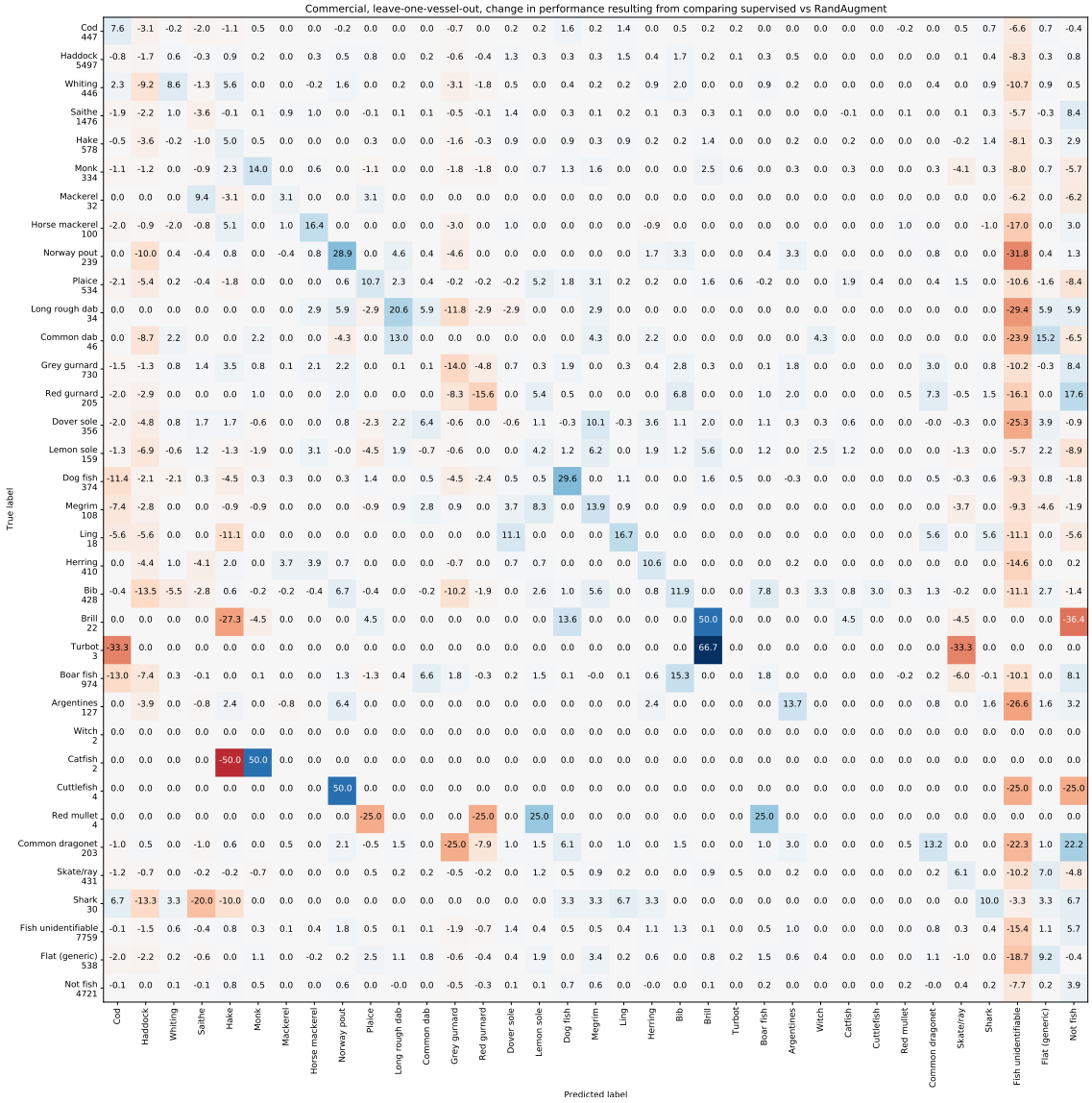


Figure 7.12: Confusion matrix comparison of supervised and semi-supervised learning (with RandAugment) results from commercial samples, exploring the inter-vessel domain gap with leave-one-vessel-out cross-validation.

Train on research and test on commercial samples

As stated in Sections 7.3.1 and 7.6.4 the videos captured on the research vessel feature fish of a single species, eliminating the need for the time consuming species annotation process. Ideally one would prefer to minimise the annotation workload by training on research vessel samples, while achieving good performance on commercial samples. This presents a larger domain gap problem than in Section 7.6.5 above. As shown in Table 7.4 the footage from the research vessel covers only 13 species. In order to make a fair comparison with the 4-fold cross-validation commercial to commercial results discussed above in Section 7.6.5 we re-compute the mean class accuracy for those results by excluding classes for which the research footage contains no training examples.

Considering the additional domain gap due to training with samples from only one vessel the results in Table 7.7 are surprisingly good when contrasted against those in Table 7.6. As explanation we offer that the research footage contains a large number of samples in each class (the most poorly represented class is red gurnard with 44 samples, in comparison to the poorly represented classes in the commercial footage which have around 2 samples). We also note an increase in score obtained from the commercial to commercial results by considering only 13 classes.

Training approach	Research to commercial Mean class accuracy (%)	Commercial to commercial 13-class Mean class accuracy (%)
Supervised	32.2%	52.6%
Semi-supervised, CowMix	39.6%	60.5%
Semi-supervised, RandAugment	48.9%	59.2%

Table 7.7: Performance of species classification when training on research samples and testing on commercial samples.

Semi-supervised learning for species identification: discussion

The results obtained above cover three real-world scenarios that vary the magnitude of the domain gap and quantify its effect on classifier performance. Considered from the perspective of semi-supervised learning the commercial-to-commercial vessel scenario is a typical semi-supervised scenario as there is little if any domain gap between supervised and unsupervised samples. We consider the leave-one-vessel-out and research-to-commercial vessel scenarios to be domain adaptation problems due to the wider domain gaps resulting from the differences in visual appearance.

The results seen above clearly demonstrate the effectiveness of semi-supervised learning using the Mean Teacher approach driven by RandAugment in each of these challenging real-world scenarios, across all three scenarios. We therefore choose this approach for training our species classifier.

7.7 Fish tracking

The segmentation and species classifier components discussed thus far are sufficient to count and identify the species of fish in single images. Counting fish within a video requires the use

of an object tracker to associate presentations of an individual fish across multiple frames with one another. This ensures that each individual fish is counted once, rather than once for each frame in which it appears.

7.7.1 Choice of approach

A variety of approaches have been proposed for object tracking in the literature, some of which are discussed in Section 2.12. For our purposes, the training data requirements of an algorithm are a very important factor. Some early hand-designed tracking approaches either do not require training or like KCF (kernelized correlation filters; Henriques et al. [2014]) are trained on-line when the tracker is initialized with the appearance of the target object. More recent deep learning based models are trained using object tracking datasets that consist of videos in which objects of interest have per-frame bounding box annotations.

Creating an object tracking training set to train a model in a supervised fashion would impose a considerable cost in terms of the annotation effort required. Given the effort required to construct the segmentation and species ID training sets described previously, we did not wish impose this additional work load on our annotators. Furthermore, our web-based annotation tool is designed for still images; modifying it for video annotation would require considerable software development effort. The performance of such a tool is also likely to be a concern as the bandwidth and latency limitations imposed by accessing the tool over the web would likely result in slow speed and a poor user experience when scrubbing through a video (playing forwards or backwards frame-by-frame, often by dragging on a time-line).

The constraints described above led us to consider the SORT [Bewley et al., 2016] and Tracktor Bergmann et al. [2019] approaches as both rely on object detection systems trained using datasets consisting of annotated still images rather than annotated video sequences. This is a natural fit given that an object detector is provided by our Mask R-CNN based instance segmentation system.

7.7.2 Fish motion estimation

Accurate camera motion estimation is often challenging in the pedestrian tracking scenes commonly explored in the object tracking literature. The apparent motion of a pedestrian from the perspective of a camera arises from two sources: the *non-rigid* motion arising from the pedestrian walking through the scene and the *rigid* motion (see Section 2.12) resulting from the motion of the camera. Modelling the camera motion and the 3D scene geometry provides a reliable estimate of the *rigid* motion. Considering this separately allows a simple motion model such as a Kalman filter [Kalman, 1960] to model the *non-rigid* motion more accurately.

In a similar fashion the motion of individual fish in our CCTV footage can be modelled as the sum of two sources: *non-rigid* motion arising from fishers picking up fish and gutting and manipulating them and *rigid* motion resulting from the conveyor belt carrying the fish resting on it. Furthermore the belt motion is highly constrained in comparison to camera motion in pedestrian scenes as it is a unidirectional translation. When processing

our CCTV footage we found that estimating the belt motion using the approach described in Section 7.4.3 is sufficient to model the rigid motion of the fish. The remaining non-rigid motion arises from either fishers gutting and manipulating fish or fish that reach the end of the conveyor belt slipping off the edge into the discard chute, thus moving faster than would be predicted by the belt motion alone.

7.7.3 Our tracker

Our tracker largely adopts the SORT [Bewley et al., 2016] approach with a few modifications. Briefly, SORT is a tracking-by-detection approach in which tracking targets are modelled as bounding boxes, with their motion estimated using a Kalman filter. A tracking target state consists of the object position, area and aspect ratio, along with positional and size velocities. The linear Kalman filter is used to estimate the bounding box for every tracking target in a new frame. Detections from a Faster R-CNN object detector are matched to tracking targets by computing the pairwise IoU (intersection over union) overlap between detections and estimated boxes and applying the Hungarian algorithm Kuhn [1955]. Given our use of a Mask-RCNN based instance segmentation system, SORT is a natural fit for the task of object tracking.

Motion model

In our fishing CCTV footage the motion of fish is somewhat more constrained than in pedestrian tracking scenarios. The apparent size of pedestrians changes due to moving closer to or further from the camera, where the motion of fish in the depth direction in our CCTV footage is much more constrained. We therefore only model the position and velocity of our targets, with the size being determined by that of the bounding box surrounding the object mask.

We found that using KCF (kernelized correlation filters) to refine motion estimates improved the accuracy of our final discard counts. The KCF algorithm estimates the motion of an object when provided as input two image regions surrounding the object that are extracted from two subsequent frames. A typical implementation extracts image crops from the same position in both frames and use KCF to estimate the object motion. In contrast we use our motion models prediction of the object position in the later frame and extract a corresponding crop and use KCF to estimate the residual motion observed in the video.

When processing a new frame, prior to matching tracking targets with detections we estimate the position of each target in the new frame. We first integrate the targets' non-rigid velocity v_n – obtained from the Kalman filter state – over the time step to estimate its non-rigid offset $\Delta p_n = v_n \Delta t$. This is summed with the rigid belt offset Δp_r to obtain the total offset $\Delta p_t = \Delta p_r + \Delta p_n$. We offset the bounding box in the current frame b_0 with the offset Δp_t to estimate the position of the bounding box in the new frame b_1. We then estimate the residual offset Δp_d using the KCF algorithm, given image regions extracted from the current and new frame corresponding to b_0 and b_1 respectively. We compute an observed non-rigid velocity $v_{obs} = \frac{\Delta p_n + \Delta p_d}{\Delta t}$ and update the Kalman filter state to estimate

the position of the object in the new frame, given previous estimate of the position and velocity and the observed velocity v_{obs}.

The original SORT [Bewley et al., 2016] algorithm stops tracking an object if it is absent for one or more frames due to not being detected, as 'the constant velocity model is a poor predictor for the true dynamics' of their problem domain and they do not attempt object re-identification. In contrast we allow fish to re-acquired by the tracker after being absent for a maximum of η_m frames (see Section 7.8.5 for a description of the process used to choose a value). This allows fish to be missing from the list of detections for brief periods of time, which can occur due to being briefly obscured by the actions of fishers or detection failures.

We obtained a small improvement in discard count accuracy by not updating the motion model for a tracked target when no detection is matched with it. Missed detections occur due to mis-predictions by Mask R-CNN or due to the fish being temporarily obscured. Manual observations revealed that in such cases the Newtonian motion modelled by the Kalman filter caused the predicted position of a tracked fish to move with a constant velocity from the time of its last successful detection. The fish in our surveillance videos tend to come to a fairly abrupt stop when they are slowed by friction or collisions with the conveyor belt or other fish. A possible future improvement to our motion model would be to decay the velocity of missing targets.

In Section 7.8.5 we will describe the hyper-parameter optimization process used to tune the parameters of our discard counter to maximize its performance on our validation data. In addition to the hyper-parameters that we present later, we tested each component of our tracker. For motion refinement we compared four approaches: no motion refinement; kernelized correlation filters (KCF); finding maximum correlation within a search window; and finding minimum L2 pixel content distance within a search window, with KCF yielding the best performance. Similarly for the motion model we compared using a Kalman filter to using a position only motion model that uses only a kernelized correlation filter to search for the tracked object at its last known position, with a belt motion offset applied. The Kalman filter based motion model yielded mildly superior performance.

Early experiments with object tracking indicated the challenges of accurately modelling the motion of fish in our CCTV footage. We found that modelling acceleration with the Kalman filter negatively impacted performance. In instances when a fish was temporarily obscured, a velocity-only Kalman filter resulted in the estimated position of the hidden fish continuing to move with a billiard ball-like trajectory, usually resulting in the fish failing to be re-identified when it became visible again. To counter this we introduced a *momentum scale* parameter that scales the velocity of the Kalman filter state on each frame – implemented by placing this value in the lower two diagonal elements of the state transition matrix – in effect simulating friction. Given that a fish tends to come to a fairly abrupt stop when a fisher drops it onto the belt after manipulating it, we found that a momentum scale of 0.001 worked well. Our Kalman filter therefore applies the effect of the velocity of the fish to its position for one frame, after which the velocity is largely damped to zero. This explains the similar performance obtained from the position only motion model.

Matching detections to targets

Prior approaches such as SORT [Bewley et al., 2016] match detections to targets by applying the Hungarian algorithm to the pairwise IoU between the target and detection bounding boxes. We use a similar approach, except that we utilize the masks predicted by Mask R-CNN, matching by mask IoU score. Any match for which the mask IoU score is lower than the threshold τ_m (default 10%) is discarded.

For each tracked fish we maintain an exponential moving average of the masks corresponding to its detections. Upon being matched with a detection with corresponding mask m_d, the target mask m_t is updated to $m'_t = \alpha_m m_t + (1 - \alpha_m) m_d$ where α_m is the mask EMA factor, with a default value of 0.75.

7.7.4 Unsuccessful tracking approaches

We also experimented with two tracking approaches that we ultimately abandoned.

Tracktor

The Tracktor object tracking approach [Bergmann et al., 2019] innovatively re-uses the R-CNN classification and bounding box refinement head of the Faster R-CNN or Mask R-CNN detector model to perform inter-frame tracking (see Section 2.12). While appealing, we found that it resulted in identity switches when multiple fish are in close proximity. We hypothesize that this is because the R-CNN head was trained for refining the object proposals generated by the RPN part of the object detection network. When given the bounding box of a tracked fish from the previous frame the R-CNN head chooses to adjust the detection bounding box to fit the individual fish that elicits the strongest response from the R-CNN head. When the strongest response arises from a different individual fish the identity is switched. We suggest that training an R-CNN head for accurate identity preservation is theoretically feasible, provided that a labelling object tracking dataset is available.

Fish re-identification

We also experimented with an unsupervised fish re-identification scheme similar to [Karthik et al., 2020]. They train a re-identification model that can be used to predict whether two tracklets generated by SORT arose from the same object, joining them into a single track if so.

The approach of Karthik et al. [2020] starts by using SORT to generate noisy tracklets (see Section 2.12). A re-identification model is trained by assigning a label to each tracklet and training a network using cross-entropy loss. The model will associate presentations from the same object more strongly with one another than from different objects. Our noisy tracklet generation process yielded millions of tracklets, which would require a multi-million class classifier. To avoid this we adapted the self-supervised contrastive approach of He et al. [2020] (our approach bore similarities to that of Azizi et al. [2021]). Images arising from the same tracklet are considered to be positive pairs, while images arising from a pair of tracklets from different videos, a pair of tracklets from chronologically distant parts of the

Figure 7.13: The discard region of vessel A shown on the left in red.

same video or from tracklets from spatially separate parts of the same frame are considered considered to be negative pairs.

While we were able to train a re-identification model, we found that its predictions were not sufficiently accurate to improve tracking performance. Visualizing the tracklets that it chose to join indicated that it made a larger number of erroneous matches, hence our Kalman filter based approach gave superior results.

7.8 Discard counter and evaluation

Our discard counter application combines the previously discussed components – instance segmentation, species classification and object tracking – with a final discard detection step into an application that processes an input video and quantifies the discarded fish.

First we will discuss the method employed by our discard counting system, followed by a description of its output. We follow this with a discussion of the format of ground truths produced by expert observers at Marine Scotland Science and Cefas and the mapping required to account for the small differences in species used in our training data and in the ground truth data. We will then discuss our evaluation metrics and how we compare the output of our system with the ground truth in order to compute them. Finally we will present our results.

7.8.1 Discard detection and application output

The lens distortion (Section 7.4.1) and belt perspective warp (Section 7.4.2) parameters are inverted in order to generate a mask that identifies the discard region that lies beyond the edge of the belt, covering the discard chute (see Figure 7.13 for an example). Our motion model – consisting of belt motion estimation, a Kalman filter and KCF as described above – predicts an inter-frame motion offset that for each target. Applying this offset may move the target mask m_t such that part of it lies within the discard region of the image. The total area of the mask that moves into the discard region is computed and accumulated on

Figure 7.14: A frame from the output video generated by the discard counter, applied to one of the test videos from vessel F. The predicted mask for each fish are used to highlight them with a partial opacity fill in a randomly chosen pastel colour. Fish that will ultimately be predicted to be discarded by our system are outlined in red, otherwise they are predicted to be retained and are outlined in yellow.

a per-frame basis. When the ratio of the accumulated discarded area to the total mask area rises above a discard threshold τ_z – with a default value of $^2/_3$ – the fish is considered to have been discarded, provided that it was detected for a minimum of η_z frames. Tracked objects detected that for less than η_z frames (default: 3) that cross into the discard region could arise from false detections, so they do not contribute to the discard count.

Fish that are processed by the fishers and retained for landing at port typically disappear from view while not in proximity to the discard chute. As they do not cross into the discard region they do not get counted as discards, as is desired. We also note that the unpredictable motion observed as a fisher rapidly manipulates and guts fish that are to be retained frequently result in them being lost by the tracker and re-detected as new targets. As these lost tracks do not cross into the discard region, they too are not counted. The end goal of by-catch quantification allows us to ignore such errors as they have little effect on the final output of the system.

When a fish is considered to be discarded, we use a species classifier to predict its species. It would be tempting to apply the classifier to an image of the fish extracted from the most recent frame in which it was visible. Doing this however would likely lead to poor classification accuracy as the fish will be in the process of falling into the discard chute. It will therefore likely be partially visible, have poor definition due to motion blur or be seen at

an orientation that obscures discriminative features, or some combination of the above. To improve accuracy we therefore average the predictions arising from images extracted from multiple video frames. Recall that our species classifiers are trained using images of individual fish that have been extracted from CCTV footage by orienting, scaling, positioning and masking them as described in Section 7.6.1. We therefore need to follow the same procedure at inference time. For each fish detected in each frame we extract an aligned (rotated to lie horizontally and scaled to a uniform size) image of the fish, using its predicted segmentation mask to estimate orientation and scale. After associating detections with tracked fish (see Section 7.7.3), the extracted images area added to a list of images that is maintained for each tracked fish. To reduce memory requirements, we retain at most 50 images of each fish, choosing those images with the largest on screen area, computed by counting the on-screen pixels covered by its segmentation mask. We chose to retain the images with the largest area as those presentations of the fish would have a higher resolution and would be less likely to be occluded, thus giving the species classifier a better image to classify. When a fish is determined to be a discard, we use the species classifier to obtain species predictions for all of the images in its image list. We obtain the final species prediction using a weighted average of the resulting probability vectors, using the on-screen area – computed from the segmentation mask – as the weight. Only classifying images of discards improves the speed of the system by avoiding the computational cost of using the species classifier unnecessarily. A discard record consisting of the predicted species and the discard time relative to the start of the video is appended to a discard log. Detections that are classified by the species classifier as either *unidentifiable fish* or *not fish* are recorded in the discard log as such. These classes must be treated specially by down-stream applications for the purpose of evaluation or final use.

Our discard counter application produces four outputs: the discard log stored in a CSV file; a human readable text log file that records each discard followed by a per-species summary at the end; the per-frame estimate of the motion of the conveyor belt; and optionally visualisation data that an be used to render a video that illustrates the fish counting process. The rendered visualisation outlines every detected fish and highlights those that are discarded, optionally showing the predicted species. A cumulative discard count is displayed in the top left of the video. Please see Figure 7.14 for an example.

7.8.2 Discard count ground truth data

Expert observers at Marine Scotland Science and CEFAS manually analysed the validation videos from vessels A, B, C, D, E and F (see Table 7.1) resulting in ground truth discard counts for each video. The validation videos were used for hyper-parameter tuning and each video was analysed by only one expert observer. The testing videos – used for final evaluation – were analysed by multiple expert observers, allowing us to evaluate the accuracy of our discard counting software in comparison with the average of multiple observers and in light of inter-observer variability. The ground truth data that we received from the expert analysts for a given video takes the form of a spreadsheet in which each row corresponds to a discarded fish, providing its species and the time at which it was discarded.

Vessel A, B, C & D ground truth species	Training species	Penalty
Anglerfish	Monk	1
Argentines	Argentines	1
Atlantic herring	Herring	1
Atlantic Horse Mackerel	Horse mackerel	1
Atlantic Mackerel	Mackerel	1
Blue Skate	Skate/ray	1
Blue Whiting	–	
Catfish	–	
Cod	Cod	1
Common dab	Common dab	1
Cuttlefish	Cuttlefish	1
Common Squids nei	Squid	1
Dogfish Sharks nei / Dogfish	Dog fish	1
Edible crab	Brown Crab	1
European Hake	Hake	1
European Plaice	Plaice	1
Flatfishes nei / Flatfish (generic)	Flat (generic)	1
Gurnards nei / Gurnard (generic)	Grey gurnard	1
Gurnards nei / Gurnard (generic)	Red gurnard	1
Red gurnard	Red gurnard	1
Gurnards nei / Gurnard (generic)	Gurnard (generic)	1
Haddock	Haddock	1
Hake	Hake	1
Herring	Herring	1
Horse Mackerel	Horse Mackerel	1
Lemon Sole	Lemon sole	1
Ling	Ling	1
Long rough dab	Long rough dab	1
Mackerel	Mackerel	1
Megrims nei	Megrim	1
Monkfish	Monk	1
Morays	–	
Norway Haddock	Norway haddock	1
Norway Lobster	European lobster	2
Norway Pout	Norway pout	1
Norway Redfish	Norway haddock	1
Octopus	–	
Plaice	Plaice	1
Poor Cod	–	
Raja rays nei	Skate/ray	1
Rays and Skates nei / Rays and Skates	Skate/ray	1
Red Stone Crab	Brown Crab	2
Saithe	Saithe	1
Squid	Squid	1
Tusked Goby	–	
Tuskfishes nei / Tuskfish	–	
Whiting	Whiting	1
Witch Flounder	Witch	1
Unidentifiable	Fish unidentifiable	1

Table 7.8: Species class mapping from ground truth classes used for vessels A, B, C and D to the set of classes used in the training set. Blank rows indicate that no training class was matched to the given ground truth class. Multiple ground truth species separated by slashes (/) indicate that slightly different species names were used by our expert observers in the validation and test set ground truths.

Vessel E & F ground truth species	Training species	Penalty
Angler Fishes	Monk	1
Atlantic Bobtail	Squid	1
Atlantic Cod	Cod	1
Bib (Pout-Whiting)	Bib	1
Blonde Ray	Skate/ray	1
Boar Fish	Boar fish	1
Common Dragonet	Common dragonet	1
Cuttlefish (With Cuttlebone)	–	
Cuckoo Ray	Skate/ray	1
Dab	Common dab	1
Edible Crab	Brown Crab	1
Epibenthic Mix Unidentified	Fish unidentifiable	1
European Common Squid	Squid	1
European Conger Eel	Conger eel	1
European Hake	Hake	1
European Lobster	European lobster	1
European Mackerel	Mackerel	1
European Plaice	Plaice	1
Fish Without Jaws	Fish unidentifiable	1
Flatfish	Flat (generic)	1
Flounder	Flat (generic)	2
Great Scallop	–	
Greater Spider Crab	Brown Crab	2
Gurnards Indet	Gurnard (generic)	1
Gurnards Indet	Grey gurnard	1
Gurnards Indet	Red gurnard	1
Haddock	Haddock	1
Herring	Herring	1
Horse-Mackerel (Scad) / Scad (Horse Mackerel) Indet	Horse mackerel	1
John Dory	John Dory	1
Lemon Sole	Lemon sole	1
Lesser Spotted Dogfish	Dog fish	1
Lesser Weever Fish	–	
Loligo Spp	Squid	1
Long-Rough Dab (American Plaice)	Long rough dab	1
Lumpsucker	–	
Megrim	Megrim	1
Norway Pout	Norway pout	1
Nurse Hound (Greater Spotted Dogfish)	Dog fish	1
Octopus Indet	–	
Poor Cod	–	
Queen Scallop	–	
Sand Sole	–	
Sandy Ray	–	
Scaldfish	–	
Skate Indet / Skates and Rays	Skate/ray	1
Small-eyed Ray (Painted Ray)	Skate/ray	1
Sole (Dover Sole)	Dover sole	1
Sole Indet	–	
Solenette	–	
Spiny Lobster Indet	European lobster	2
Spotted Ray	Skate/ray	1
Squids and Octopi	Squid	1
Starry Smooth Hound	Dog fish	1
Striped Red Mullet	Red mullet	1
Thickback Sole	Flat (generic)	2
Thornback Ray	Skate/ray	1
Undulate (Painted) Ray	Skate/ray	1
Whiting	Whiting	1
Flatfish	Flat (generic)	1
Unidentifiable	Fish unidentifiable	1

Table 7.9: Species class mapping from ground truth classes used for vessels E and Fs to the set of classes used in the training set.

Species classes

Upon inspecting the ground truth discard counts we found that the set of classes used does not match those used during training. As a consequence, a mapping between the classes had to be established. Our class mappings from ground truth classes to training classes are given in Tables 7.8 and 7.9. Classes with identical names were trivially matched with one another. Other classes were matched due to their similarity, *e.g.* Atlantic herring (vessels A, B, C, & D ground truth) with Herring (training). Other less precise matches were made, but

with a higher penalty. The penalty value is used during our evaluation (described below) to match individual ground truth discards with predicted discards.

As mentioned in Section 7.8.1 above, discards that are assigned a non-fish species class are ignored during our evaluation. Predicted discards assigned the class *not fish* are ignored, as are ground truth discards given the *Marine litter* class.[5]

7.8.3 Matching individual ground truth and predicted discards

In Section 7.8.4 below we will describe the metrics that we use to evaluate our system. In addition to count accuracy, we need to analyse the accuracy of individual discards, as computing the individual accuracy metrics described in Section 7.8.4 requires a comparison between the predictions from our system and the ground truth for each individual discarded fish observed in a video. This requires that we match the ground truth and predicted discards with one another for comparison. A precise match would require the ground truth discard data to provide per-frame instance segmentation – or at least bounding boxes – for each discarded fish. This would allow us to ensure an exact match between ground truth and predicted discards. The additional manual labour required to provide per-frame polygonal or bounding box annotations for the ground truth makes such a precise matching an infeasible proposition. As a consequence we approximately match ground truth and predicted discards using only an estimate of their position along the travel direction of the conveyor belt – computed from discard time – and species.

We match ground truth and predicted discards such that discards of the same or compatible species (see Section 7.8.2 above) and with similar positions along the belt are more likely to be matched. If the distance between two discards is above a certain threshold we consider that it is highly unlikely that the discard records will correspond to the same individual fish, so they will not be matched. We estimate the position of a discard along the belt from its discard timestamp using the estimate of the belt motion – as computed in Section 7.4.3 – by converting the timestamp to a frame number and obtaining the cumulative belt motion for that frame. Further down we represent this conversion by the function $\rho(\cdot)$. We chose this approach in preference to matching discards by their discard timestamp alone – a solution we explored earlier – as the accuracy of a timestamp based approach would be hampered by the numerous periods in our CCTV videos where the belt remains stationary for significant periods of time. Under the assumption that human observers and our software both determine a fish to be a discard once it crosses a physical threshold on the belt adjacent to the discard chute, any discrepancy between the thresholds used by the two – or between the thresholds used by different human observers – could result in a large difference between estimated discard timestamps for a given fish should the belt remain stationary for a long period of time at the right moment.

Both the ground truth data and the discard logs generated by our counting software provide a species and discard timestamp for each discarded fish. We note that the discard timestamps in the ground truth take the form of the date and time at which the discard occurred, whereas our predictions give the time relative to the start of the video. To support direct

[5]The expert observers quantified litter with a view to exploring this in a future project.

comparison between ground truth and predicted discard timestamps we compute ground truth discard timestamps relative to the start of the video by subtracting the video start date and time.

Our matching process matches a ground truth discard log consisting of M discards with N predicted discards. The entry in the ground truth log for discard i consists of species class c_i and time s_i where $i < M$. Similarly, entry in the prediction log for discard j consists of species class d_j and time t_j where $j < N$. As noted earlier, the set of species classes used in the ground truth counts differs from the set of classes used for training and prediction. The species classes c_i and d_j are represented as integers that index into the ground truth and training class sets respectively. The discard times s_i and t_j are converted to positions $p_i = \rho(s_i)$ and $q_j = \rho(t_j)$.

Our approach constructs a $M \times N$ cost matrix A that is used to match ground truth discards to predictions using the Hungarian algorithm [Kuhn, 1955]. Each entry $A_{i,j}$ gives the cost of matching the ground truth i with the predicted discard j. We will now discuss how this cost matrix is computed.

We expect that there may often be a discrepancy between the discard positions for a given fish reported in the ground truth and predictions, so we encourage ground truth and predicted discards to match if their discard positions lie within a window of w_{max} metres; $|p_i - q_j| < w_{max}$, where w_{max} defaults to 0.5 metres. A window size of 0.5 metres allows the predicted and ground truth discard positions – and therefore times – to differ, accounting for differences that are likely to exist between when our algorithm and when a human observer consider a fish to be discarded. For GT-prediction discard pairings whose times differ by more than w_{max}, a large *no-match* cost of 100,000 is used.

A species class match penalty function $p(c_i, d_j)$ returns the class match penalty value specified in Tables 7.8 and 7.9 for matching a ground truth discard with class c_i with a predicted discard with class d_j. For a species class pair c_i, d_j for which no penalty value is given, a default *bad match* penalty b of 10 is returned by $p(c_i, d_j)$. The penalty value multiplied by w_{max} is added to the cost, encouraging matches between GT-prediction pairs where the species classes are compatible.

Thus the cost matrix A is computed as follows:

$$A_{i,j} = \begin{cases} |p_i - q_j| + p(c_i, d_j) w_{max}, & \text{if } |p_i - q_j| \leq w_{max} \\ 10^5, & \text{otherwise} \end{cases}$$

Given the matrix A the Hungarian algorithm produces in a list of matches between the ground truth and predicted discards that we use to evaluate the accuracy of our discard counter. The construction of A prioritizes matching fish of the same class over similar discard positions.

7.8.4 Evaluation metrics

We chose metrics to evaluate the accuracy of our discard counter in terms of the outputs that would be most useful to users. Marine biologists require an accurate per-species count of the by-catch (discards) observed in a given video. We compute the accuracy of per-species counts in order to give users an impression of the reliability of the output of the system for the intended use case. We note however that predicted counts can conceal errors as false positives and false negatives can cancel one another out. We therefore also analyse the accuracy of predictions for individual discards using confusion matrices and precision and recall metrics.

Count accuracy

We measure count accuracy by computing:

$$\frac{\min(|P_c|, |T_c|)}{\max(|P_c|, |T_c|)} \tag{7.1}$$

where $|P_c|$ is the number of predicted discards of species class c and $|T_c|$ is the number of ground truth discards of class c. This measure will penalise the accuracy score in cases where our system either under or over counts fish of a given species.

Extended confusion matrices

We present extended confusion matrices that quantify the predictions and mis-predictions on a per-species basis. The extensions consist of an additional row (in grey) at the bottom that quantifies false positives (predicted discards that were not matched to a ground truth discard) and an additional column (also in grey) on the right quantifies false negatives (ground truth discards that were not detected).

Species summary tables

The per-species summary tables present count accuracy, precision and recall values for each species, with row per species. We generate one summary table for each vessel and a final per-species summary table for all vessels combined. We provide the following metrics:

- True and predicted counts measure the true and predicted number of discards of a given species over all videos from the vessel.
- The *vessel count accuracy* column provides the accuracy of the count over all footage from the vessel for a given species, computed using Equation 7.1.
- The *vessel precision* and *vessel recall* columns give the precision and recall considering matched individual discards. Computing these metrics requires the number of true positives, false positives and false negatives. Ground truth and predicted discards that are matched and are of the same species are viewed as true positives, while ground truth discards that are not matched (detected) or are wrongly classified are viewed as false negatives. Predicted discards that are not matched to a ground truth discard or

are matched to one of a different class are viewed as false positives. The *vessel recall* will match the corresponding entry on the diagonal of the confusion matrix (*e.g.* in Figure 7.15).

- The *video count accuracy*, *video precision* and *video recall* columns compute the same metrics separately for each video and average them, showing the expected performance for a video, rather than for fish over all videos from a vessel.

At the bottom of the table we add two rows that give aggregate statistics over all species:

- *All Species*: the *true count*, *predicted count* and *count accuracy* columns provide the total counts and count accuracies of all fish without considering species. The *precision* and *recall* are weighted according to class frequency; they provide the precision and recall values one can expect for any given discard. They are computed by summing the number of true positives, false positives and false negatives over all species classes before computing these metrics. Note that the total true positive count is the number of matched predicted and ground truth discards from that vessel where the species match.
- *Average*: *true count* and *predicted count* are the number of ground truth and predicted discards averaged over species class. *Count accuracy* is the average of the per-species count accuracies, or the average accuracy one can expect for any species. *Precision* and *recall* are the respective values averaged over species classes; they provide the expected precision and recall values one can expect within a class.

Video summary tables

The per-video summary tables present the metrics from the *All Species* row of the species summary table for each video. The true and predicted counts are total counts across all species for the given video and the count accuracy metric is computed from these values. The precision and recall are computed from the true positive, false positive and false negative counts for that video, as for the *vessel precision* and *vessel recall* metrics from the *All Species* row of the per-species summary tables. A final *average* row averages the values of the metrics over the videos, giving the average performance for any given video.

7.8.5 Hyper-parameter optimization

As stated in Section 7.8.2 the validation set videos were used for hyper-parameter optimization. The optimized hyper-parameter values were then used to obtain predictions for the videos in the test set.

We opted for a manual optimization process in which we tuned each hyper-parameter in turn. For each hyper-parameter we tested a range of manually chosen values, selecting the value that resulted in the best outcome.

We developed a fitness score that blends metrics indicative of practical real-world performance and used it to guide our optimization process. Our fitness score is a weighted average of the following metrics taken from a species summary table (see above) covering all vessels:

- The *video count accuracy* averaged over all species; as above the count accuracy is computed separately for a given species for each video. The video count accuracies are then averaged to given a video count accuracy for that species. The per-species video count accuracies are then averaged.
- The *count accuracy* averaged over all species; the count accuracy is computed for a given species considering discards from *all* videos. The per-species count accuracies are then averaged.
- F1-score; the per-species *precision* and *recall* values are computed, from which a per-species *F1-score* is computed. The per-species F1-scores are then averaged.
- video F1-score; the per-species *precision* and *recall* values are computed for each video, from which a per-species per-video *F1-score* is computed. The F1-scores for a given species are averaged to obtain the average F1-score that would be expected for any video for that species. The per-species average F1-scores are then averaged.

The fitness score is computed by giving the *video count accuracy* a weight of 2 and the other metrics a weight of 1.

We optimized the following hyper-parameters:

- *Detection confidence threshold* τ_d*:* the confidence threshold used for inference with Mask R-CNN (see Section 7.5.2). Optimized value: 0.7.
- *Mask NMS threshold* τ_n*:* the mask NMS threshold (see Section 7.5.2). Optimized value: 0.95.
- *Tracker mask IoU threshold* τ_m*:* the minimum IoU for which the mask predicted for a detection in the current frame can be matched a tracked object (see Section 7.7.3). Optimized value: 0.1.
- *Mask EMA factor* α_m*:* the EMA factor used to update the mask of a tracked object (see Section 7.7.3). Optimized value: 0.75.
- *Maximum number of absent frames* η_m*:* the maximum number of frames for which a fish can be absent (not detected) before the tracker stops tracking it (see Section 7.7.3). Optimized value: 7.
- *Discard area threshold* τ_z*:* the proportion of the fish that must cross into the discard region for a fish to be counted as a discard (see Section 7.8). Optimized value: 0.333.
- *Minimum discard detection frame count* η_z*:* the minimum count of frames in which a fish must have been detected for it to be counted as a discard (see Section 7.7.3). Optimized value: 6.

7.8.6 Results

We evaluated our discard counter while using a segmentation model and species classification model trained on a snapshot of the data generated in August 2021. The classifier was trained in a semi-supervised fashion using RandAugment as described in Section 7.6.3,

using training data drawn from all vessels using the first of four cross-validation folds as described in Section 7.6.5. We note here that the videos for which ground truth discard counts were manually prepared do not have *any* segmentation or species ID annotations, thus we can use all images for which we have segmentation and species annotations for training without any risk of train-test contamination.

We present our results in three forms: an extended confusion matrix, a vessel summary table and a video summary table as outlined in Section 7.8.4.

The evaluation results obtained after hyper-parameter optimization on the validation videos are presented in Appendix B Section B.1.

Evaluation using multiple observers

The availability of ground truth counts from multiple observers affords us the opportunity to compare the predictions from our system against the average of multiple observers. We then compare the performance of our model with the inter-observer variability; the performance of the observers against one another.

The performance metrics described in Section 7.8.4 are obtained by first computing an un-normalized extended confusion matrix, with false positives and false negatives in an extra row and column respectively. The values in the un-normalized confusion matrices are counts and are not divided by the total number of samples in each class (the sum of each row). From this confusion matrix we can compute precision and recall metrics.

We assess the performance of our system with respect to multiple observers by comparing the predictions of our system to the average of the expert observers. For a set of expert observers S we compute the un-normalized confusion matrix C_o for each observer o in S. Each one asses our system with respect to each individual observer. We then compute the per-element mean confusion matrix $C = \frac{\sum_o^S C_{o;i,j}}{|S|}$. The resulting mean confusion matrix may have a fractional count for species where observers disagreed on the number of discards seen. This mean confusion matrix can then be used to compute precision and recall metrics.

Furthermore we assess the performance of our system in light of the variability between expert observers. We quantify inter-observer variability by computing the average performance of any given expert observer in comparison to the other observers in the study. For each expert observer $p \in S$ we compute the the mean confusion matrix B_p that is the per-element mean of $D_{p,q}$ for each $q \in S, q \neq p$. $D_{p,q}$ is an un-normalized confusion matrix that assesses the observations from observer p against those of observer q. From this we can compute performance metrics – normalized confusion matrix, precision and recall – for the observer p. We then compute the average of the performance by computing the mean of the performance metrics for all observers $p \in S$.

Results: test videos

Extended confusion matrices break down the correct and incorrect predictions on a per-species basis for vessels A, B, C, D, E and F in Figures 7.15, 7.16, 7.17, 7.18, 7.19 and 7.20

respectively. The confusion matrices compare the performance of our system comparison with that of the average of the observers.

The per-species performance is summarised – along with count accuracies – for vessels A-F in Tables 7.10, 7.11, 7.12, 7.13, 7.14 and 7.15 and for all vessels in Table 7.16. The per-video performance is summarised in Table 7.17. Once again, we compare the performance of our system with that of the average of the observers.

Results: out-of-training-distribution species

We would like to draw attention to our extremely strict approach for evaluating the performance of our discard counter. The orange rows in the confusion matrices in Figures 7.15, 7.16, 7.17, 7.18, 7.19 and 7.20 and the red rows in Tables 7.10, 7.11, 7.12, 7.13, 7.14, 7.15 and 7.16 correspond to species that were present in the test set but *not* in the training set. As a consequence our discard counter was unable to identify fish belonging to these species, hence receiving a zero score for all performance metrics in these rows. We chose this approach as it measures the real-world performance of our system on our test set, with the caveat that it could be seen to be overly strict. We chose this as an alternative to either excluding these fish from the evaluation (deciding which predicted discards correspond to the out-of-training-distribution ground truth individuals would not necessarily be accurate) or giving positive score for any species prediction for these individuals.

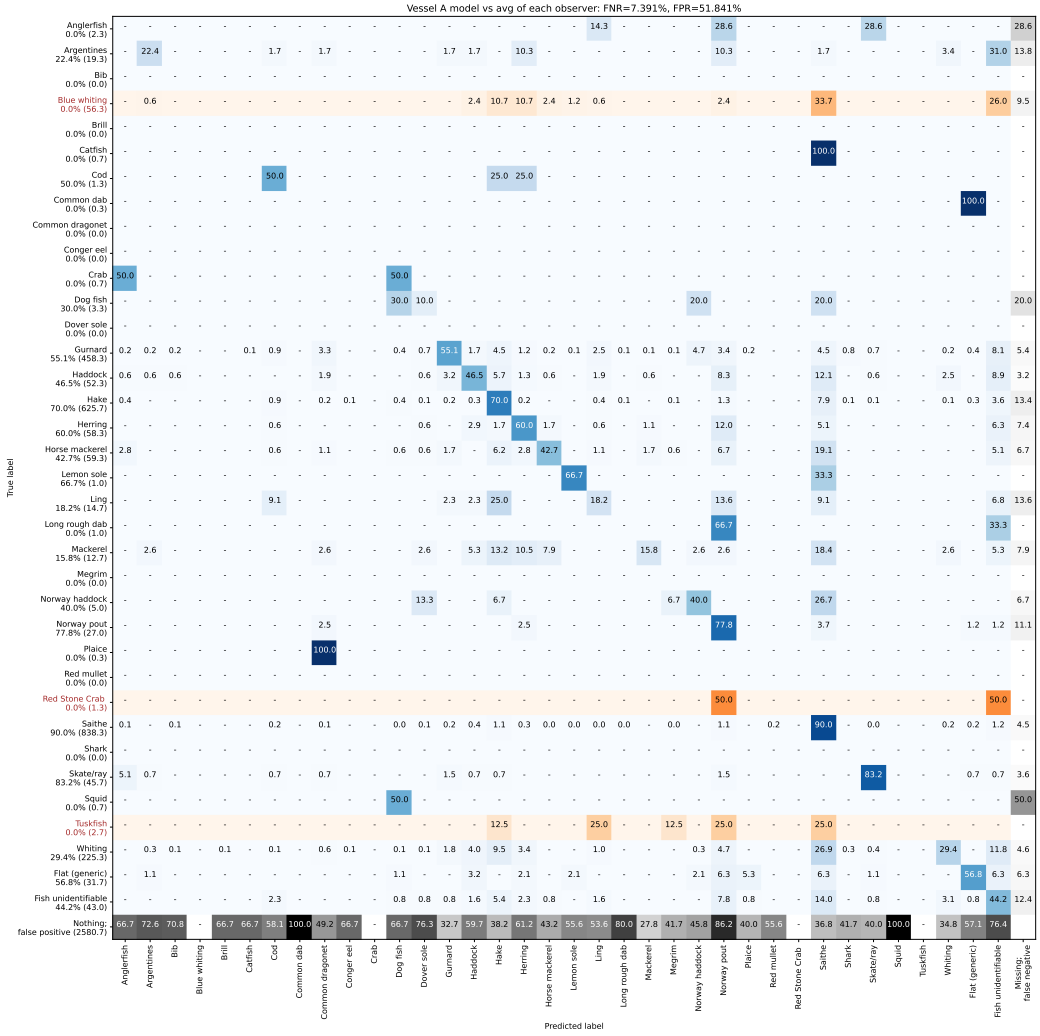


Figure 7.15: Discard counter evaluation for vessel A presented as an extended confusion matrix. The performance of the discard counter is evaluated against the average of multiple observers. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions (non-zero values along the diagonal). The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from detection and tracking errors.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	2.3	26.0	18.15%	0.00%	0.00%	0.00%	8.97%	0.00%	0.00%	0.00%
Argentines	127	19.3	28.0	70.30%	16.40%	18.24%	16.56%	69.05%	15.48%	22.41%	18.31%
Bib	428	0.0	8.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Blue whiting	0	56.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Brill	22	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Catfish	2	0.7	1.0	66.67%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Cod	447	1.3	39.0	5.33%	2.67%	50.00%	4.21%	3.42%	1.71%	50.00%	3.31%
Common dab	46	0.3	2.0	16.67%	0.00%	0.00%	0.00%	16.67%	0.00%	0.00%	0.00%
Common dragonet	203	0.0	44.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Conger eel	1	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Crab	0	0.7	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Dog fish	376	3.3	23.0	7.41%	2.22%	30.00%	3.24%	14.49%	4.35%	30.00%	7.59%
Dover sole	360	0.0	31.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	458.3	391.0	68.50%	58.07%	40.87%	43.51%	85.31%	64.54%	55.05%	59.42%
Haddock	5498	52.3	129.0	39.97%	17.16%	29.79%	20.70%	40.57%	18.86%	46.50%	26.84%
Hake	578	625.7	828.0	55.65%	32.55%	64.71%	38.91%	75.56%	52.90%	70.01%	60.26%
Herring	410	58.3	170.0	23.94%	10.40%	48.95%	15.87%	34.31%	20.59%	60.00%	30.66%
Horse mackerel	100	59.3	54.0	42.85%	25.25%	24.32%	22.27%	91.01%	46.91%	42.70%	44.71%
Lemon sole	162	1.0	6.0	25.00%	16.67%	75.00%	22.50%	16.67%	11.11%	66.67%	19.05%
Ling	18	14.7	51.0	35.77%	4.84%	13.93%	6.86%	28.76%	5.23%	18.18%	8.12%
Long rough dab	34	1.0	5.0	20.83%	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%	0.00%
Mackerel	32	12.7	6.0	42.36%	16.67%	5.56%	7.69%	47.37%	33.33%	15.79%	21.43%
Megrim	108	0.0	4.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway haddock	4	5.0	48.0	26.67%	16.67%	40.00%	14.81%	10.42%	4.17%	40.00%	7.55%
Norway pout	239	27.0	686.0	12.26%	3.15%	60.11%	5.76%	3.94%	3.06%	77.78%	5.89%
Plaice	534	0.3	5.0	0.00%	0.00%	0.00%	0.00%	6.67%	0.00%	0.00%	0.00%
Red mullet	4	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red Stone Crab	0	1.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	838.3	1489.0	44.94%	37.04%	72.04%	47.34%	56.30%	50.68%	90.02%	64.85%
Shark	30	0.0	8.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	45.7	75.0	52.61%	45.43%	91.31%	56.89%	60.89%	50.67%	83.21%	62.98%
Squid	74	0.7	1.0	0.00%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Tuskfish	0	2.7	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	225.3	112.0	54.06%	40.84%	27.80%	32.08%	49.70%	59.23%	29.44%	39.33%
Flat (generic)	538	31.7	56.0	56.28%	18.77%	39.32%	20.91%	56.55%	32.14%	56.84%	41.06%
Fish unidentifiable	7767	43.0	645.0	6.97%	2.87%	45.86%	5.32%	6.67%	2.95%	44.19%	5.52%
Any species	–	2588.7	4978.0	51.62%	28.42%	57.25%	36.93%	52.00%	34.26%	65.88%	45.07%
Average	–	2588.7	4978.0	22.03%	11.49%	27.78%	10.71%	26.02%	14.93%	32.10%	14.64%

Table 7.10: Discard counter performance species summary for vessel A. The performance of the discard counter is evaluated against the average of multiple observers. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

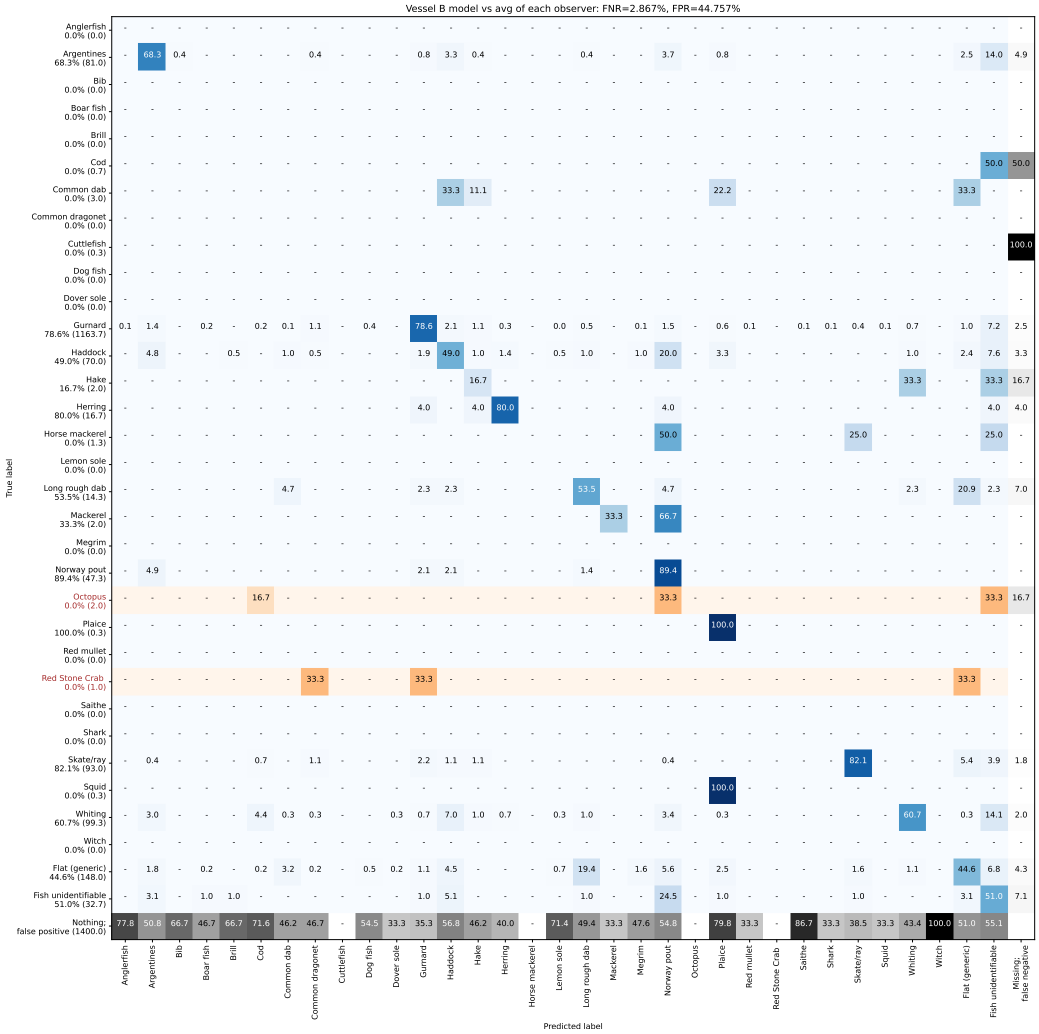


Figure 7.16: Discard counter evaluation for vessel B presented as an extended confusion matrix. The performance of the discard counter is evaluated against the average of multiple observers. Please see the caption of Figure 7.15 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Argentines	127	81.0	172.0	47.45%	36.87%	76.90%	48.31%	47.09%	32.17%	68.31%	43.74%
Bib	428	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Boar fish	1002	0.0	5.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Brill	22	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	0.7	27.0	1.90%	0.00%	0.00%	0.00%	2.47%	0.00%	0.00%	0.00%
Common dab	46	3.0	13.0	11.11%	0.00%	0.00%	0.00%	23.08%	0.00%	0.00%	0.00%
Common dragonet	203	0.0	30.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cuttlefish	4	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Dog fish	376	0.0	11.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dover sole	360	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	1163.7	1427.0	78.81%	63.47%	68.41%	64.79%	81.55%	64.07%	78.57%	70.59%
Haddock	5498	70.0	186.0	37.43%	21.21%	59.32%	29.74%	37.63%	18.46%	49.05%	26.82%
Hake	578	2.0	31.0	4.40%	0.95%	33.33%	1.82%	6.45%	1.08%	16.67%	2.02%
Herring	410	16.7	30.0	35.21%	28.21%	70.43%	30.26%	55.56%	44.44%	80.00%	57.14%
Horse mackerel	100	1.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lemon sole	162	0.0	7.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Long rough dab	34	14.3	89.0	26.98%	13.76%	25.42%	12.87%	16.10%	8.61%	53.49%	14.84%
Mackerel	32	2.0	1.0	33.33%	66.67%	50.00%	40.00%	50.00%	66.67%	33.33%	44.44%
Megrim	108	0.0	7.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway pout	239	47.3	222.0	6.03%	5.39%	89.44%	8.29%	21.32%	19.07%	89.44%	31.44%
Octopus	0	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	534	0.3	76.0	0.56%	0.56%	100.00%	1.08%	0.44%	0.44%	100.00%	0.87%
Red mullet	4	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red Stone Crab	0	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	0.0	5.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Shark	30	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	93.0	136.0	56.58%	45.80%	83.86%	56.08%	68.38%	56.13%	82.08%	66.67%
Squid	74	0.3	1.0	0.00%	0.00%	0.00%	0.00%	33.33%	0.00%	0.00%	0.00%
Whiting	446	99.3	126.0	46.21%	24.05%	39.51%	26.66%	78.84%	47.88%	60.74%	53.55%
Witch	2	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Flat (generic)	538	148.0	187.0	40.33%	25.01%	59.42%	28.13%	79.14%	35.29%	44.59%	39.40%
Fish unidentifiable	7767	32.7	328.0	9.63%	4.80%	57.23%	8.56%	9.96%	5.08%	51.02%	9.24%
Any species	–	1779.0	3128.0	56.58%	39.42%	69.31%	50.15%	56.87%	41.18%	72.40%	52.50%
Average	–	1779.0	3128.0	13.21%	11.61%	40.66%	10.81%	18.53%	13.77%	40.36%	13.96%

Table 7.11: Discard counter performance species summary for vessel B. The performance of the discard counter is evaluated against the average of multiple observers. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

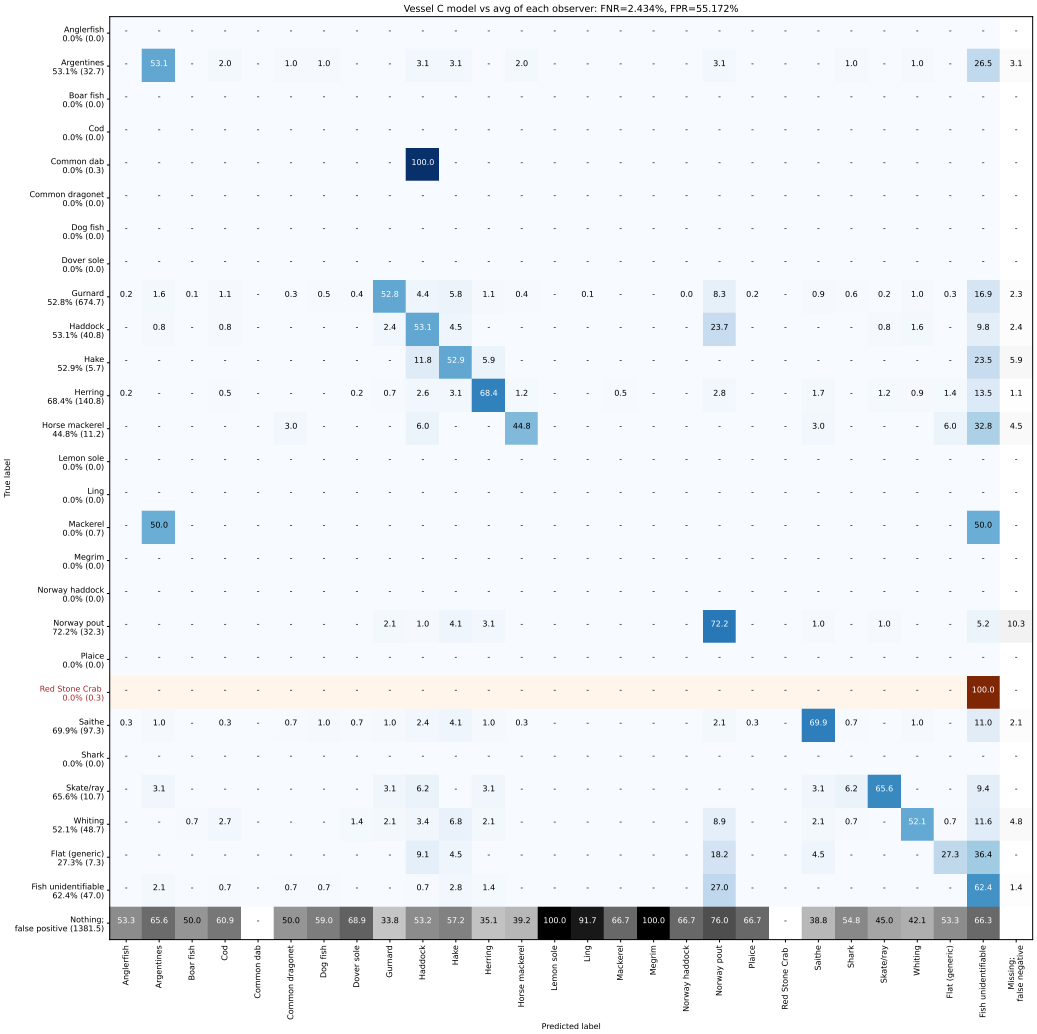


Figure 7.17: Discard counter evaluation for vessel C presented as an extended confusion matrix. The performance of the discard counter is evaluated against the average of multiple observers. Please see the caption of Figure 7.15 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	0.0	5.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Argentines	127	32.7	91.0	33.67%	20.50%	56.27%	24.32%	35.90%	19.05%	53.06%	28.03%
Boar fish	1002	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	0.0	29.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Common dab	46	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	203	0.0	8.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	0.0	13.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dover sole	360	0.0	15.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	674.7	546.0	74.24%	74.13%	53.73%	61.36%	80.93%	65.26%	52.82%	58.38%
Haddock	5498	40.8	136.0	21.20%	13.41%	47.70%	16.93%	30.02%	15.93%	53.06%	24.51%
Hake	578	5.7	139.0	5.54%	2.53%	56.67%	4.46%	4.08%	2.16%	52.94%	4.15%
Herring	410	140.8	167.0	45.10%	43.57%	61.86%	39.13%	84.33%	57.68%	68.40%	62.59%
Horse mackerel	100	11.2	17.0	23.89%	10.42%	16.13%	8.96%	65.69%	29.41%	44.78%	35.50%
Lemon sole	162	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Ling	18	0.0	8.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Mackerel	32	0.7	2.0	33.33%	0.00%	0.00%	0.00%	33.33%	0.00%	0.00%	0.00%
Megrim	108	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway haddock	4	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway pout	239	32.3	477.0	5.78%	3.45%	71.46%	6.32%	6.78%	4.89%	72.16%	9.16%
Plaice	534	0.0	6.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red Stone Crab	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	97.3	129.0	51.27%	26.28%	46.44%	31.38%	75.45%	52.71%	69.86%	60.09%
Shark	30	0.0	14.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	10.7	20.0	56.97%	37.50%	65.28%	45.61%	53.33%	35.00%	65.62%	45.65%
Whiting	446	48.7	61.0	70.76%	39.45%	46.05%	41.46%	79.78%	41.53%	52.05%	46.20%
Flat (generic)	538	7.3	15.0	32.89%	8.89%	26.26%	11.34%	48.89%	13.33%	27.27%	17.91%
Fish unidentifiable	7767	47.0	601.0	5.17%	3.40%	81.15%	6.19%	7.82%	4.88%	62.41%	9.05%
Any species	–	1150.5	2504.0	41.99%	23.32%	54.63%	32.01%	45.95%	26.14%	56.90%	35.83%
Average	–	1150.5	2504.0	17.03%	11.34%	41.93%	11.02%	22.46%	13.67%	44.96%	14.86%

Table 7.12: Discard counter performance species summary for vessel C. The performance of the discard counter is evaluated against the average of multiple observers. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

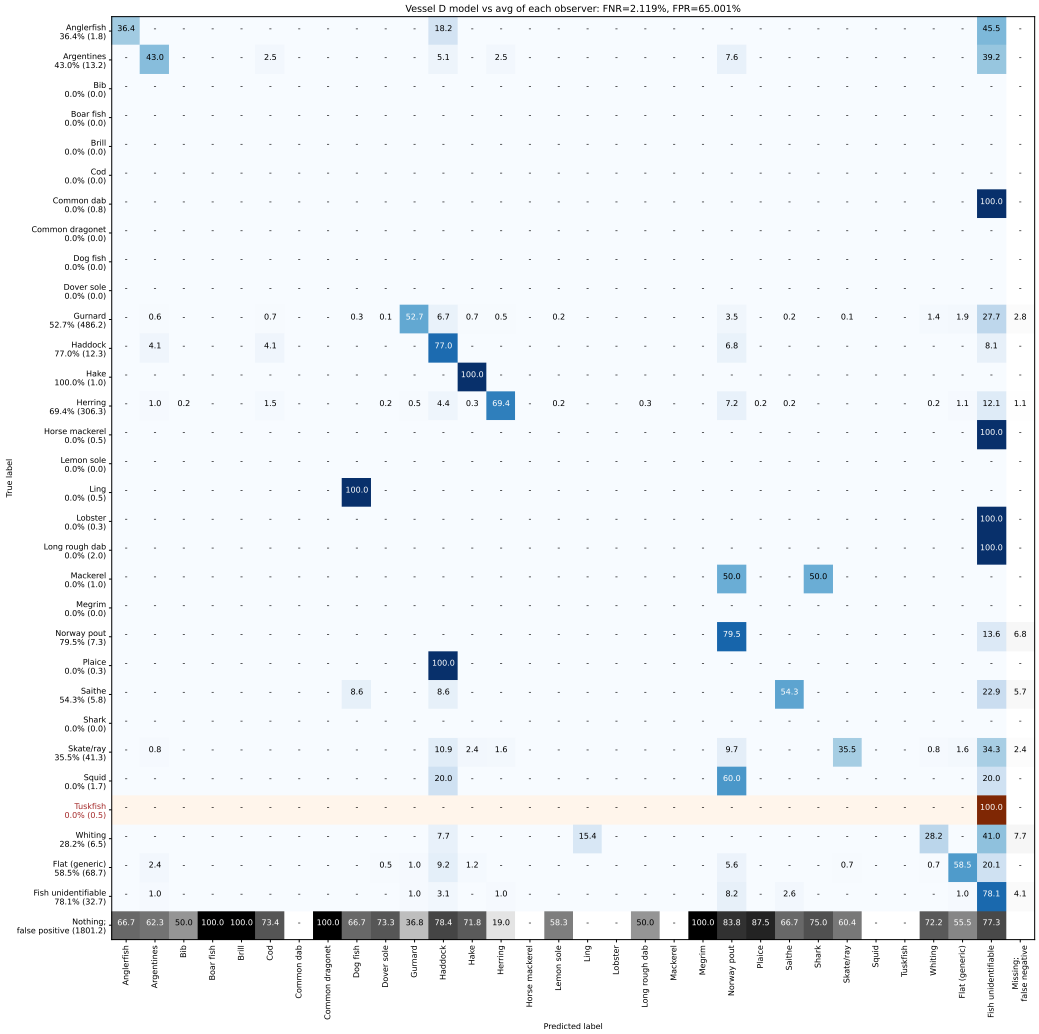


Figure 7.18: Discard counter evaluation for vessel D presented as an extended confusion matrix. The performance of the discard counter is evaluated against the average of multiple observers. Please see the caption of Figure 7.15 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	1.8	2.0	44.44%	33.33%	33.33%	26.67%	91.67%	33.33%	36.36%	34.78%
Argentines	127	13.2	38.0	33.02%	15.08%	42.29%	21.59%	34.65%	14.91%	43.04%	22.15%
Bib	428	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Boar fish	1002	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Brill	22	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	0.0	32.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Common dab	46	0.8	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	203	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	0.0	7.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dover sole	360	0.0	5.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	486.2	409.0	75.21%	66.17%	57.50%	59.97%	84.13%	62.59%	52.66%	57.20%
Haddock	5498	12.3	324.0	4.14%	3.22%	80.00%	5.87%	3.81%	2.93%	77.03%	5.65%
Hake	578	1.0	26.0	4.00%	4.00%	100.00%	6.67%	3.85%	3.85%	100.00%	7.41%
Herring	410	306.3	267.0	62.97%	54.54%	72.45%	58.48%	87.16%	79.59%	69.37%	74.13%
Horse mackerel	100	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lemon sole	162	0.0	4.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Ling	18	0.5	1.0	50.00%	0.00%	0.00%	0.00%	50.00%	0.00%	0.00%	0.00%
Lobster	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	34	2.0	2.0	100.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%
Mackerel	32	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Megrim	108	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway pout	239	7.3	363.0	1.34%	1.15%	93.18%	2.24%	2.02%	1.61%	79.55%	3.15%
Plaice	534	0.3	4.0	0.00%	0.00%	0.00%	0.00%	8.33%	0.00%	0.00%	0.00%
Saithe	1477	5.8	16.0	38.49%	22.02%	45.83%	28.45%	36.46%	19.79%	54.29%	29.01%
Shark	30	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	41.3	40.0	62.06%	50.32%	29.74%	35.20%	96.77%	36.67%	35.48%	36.07%
Squid	74	1.7	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Tuskfish	0	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	6.5	36.0	32.38%	9.76%	28.33%	13.19%	18.06%	5.09%	28.21%	8.63%
Flat (generic)	538	68.7	121.0	56.17%	35.22%	64.18%	41.54%	56.75%	33.20%	58.50%	42.36%
Fish unidentifiable	7767	32.7	1066.0	3.86%	2.90%	87.26%	5.37%	3.06%	2.39%	78.06%	4.64%
Any species	–	990.8	2771.0	34.57%	20.12%	57.91%	29.58%	35.76%	20.80%	58.18%	30.65%
Average	–	990.8	2771.0	18.33%	11.91%	34.96%	9.85%	21.83%	11.84%	33.93%	10.49%

Table 7.13: Discard counter performance species summary for vessel D. The performance of the discard counter is evaluated against the average of multiple observers. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

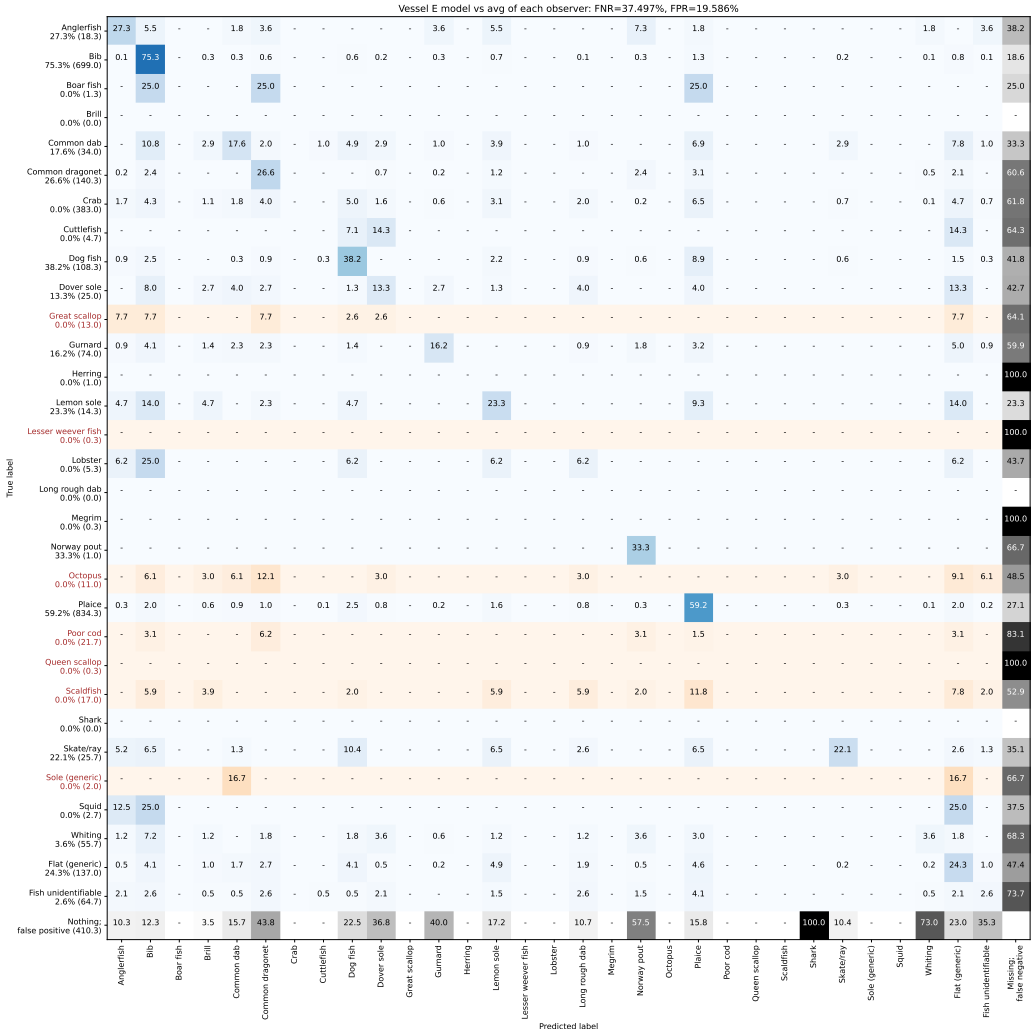


Figure 7.19: Discard counter evaluation for vessel E presented as an extended confusion matrix. The performance of the discard counter is evaluated against the average of multiple observers. Please see the caption of Figure 7.15 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	18.3	26.0	46.27%	14.22%	19.25%	12.61%	70.51%	19.23%	27.27%	22.56%
Bib	428	699.0	680.0	58.48%	31.69%	48.30%	32.20%	97.28%	77.45%	75.35%	76.38%
Boar fish	1002	1.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Brill	22	0.0	19.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Common dab	46	34.0	36.0	46.69%	20.00%	21.71%	17.40%	94.44%	16.67%	17.65%	17.14%
Common dragonet	203	140.3	143.0	46.12%	16.47%	33.86%	18.32%	98.14%	26.11%	26.60%	26.35%
Crab	0	383.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	4	4.7	2.0	19.44%	0.00%	0.00%	0.00%	42.86%	0.00%	0.00%	0.00%
Dog fish	376	108.3	129.0	50.32%	23.77%	28.54%	22.17%	83.98%	32.04%	38.15%	34.83%
Dover sole	360	25.0	39.0	38.38%	9.79%	16.00%	8.91%	64.10%	8.55%	13.33%	10.42%
Great scallop	0	13.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	6040	74.0	35.0	53.68%	40.51%	20.44%	26.26%	47.30%	34.29%	16.22%	22.02%
Herring	410	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lemon sole	162	14.3	62.0	24.75%	5.30%	22.78%	6.13%	23.12%	5.38%	23.26%	8.73%
Lesser weever fish	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	0	5.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	34	0.0	28.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Megrim	108	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Norway pout	239	1.0	40.0	4.18%	0.56%	33.33%	1.08%	2.50%	0.83%	33.33%	1.63%
Octopus	0	11.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	534	834.3	671.0	78.34%	77.53%	61.28%	67.71%	80.42%	73.62%	59.21%	65.63%
Poor cod	0	21.7	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Queen scallop	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Scaldfish	0	17.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Shark	30	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	25.7	16.0	50.56%	18.75%	15.42%	13.78%	62.34%	35.42%	22.08%	27.20%
Sole (generic)	0	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Squid	74	2.7	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	55.7	21.0	17.77%	16.67%	2.55%	3.69%	37.72%	9.52%	3.59%	5.22%
Flat (generic)	538	137.0	129.0	37.77%	33.76%	22.38%	20.22%	94.16%	25.84%	24.33%	25.06%
Fish unidentifiable	7767	64.7	17.0	22.25%	15.28%	6.29%	6.75%	26.29%	9.80%	2.58%	4.08%
Any species	–	2695.3	2095.0	72.28%	56.18%	40.79%	46.74%	77.73%	55.94%	43.48%	48.93%
Average	–	2695.3	2095.0	19.19%	18.02%	12.58%	8.30%	29.84%	20.82%	13.68%	11.20%

Table 7.14: Discard counter performance species summary for vessel E. The performance of the discard counter is evaluated against the average of multiple observers. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

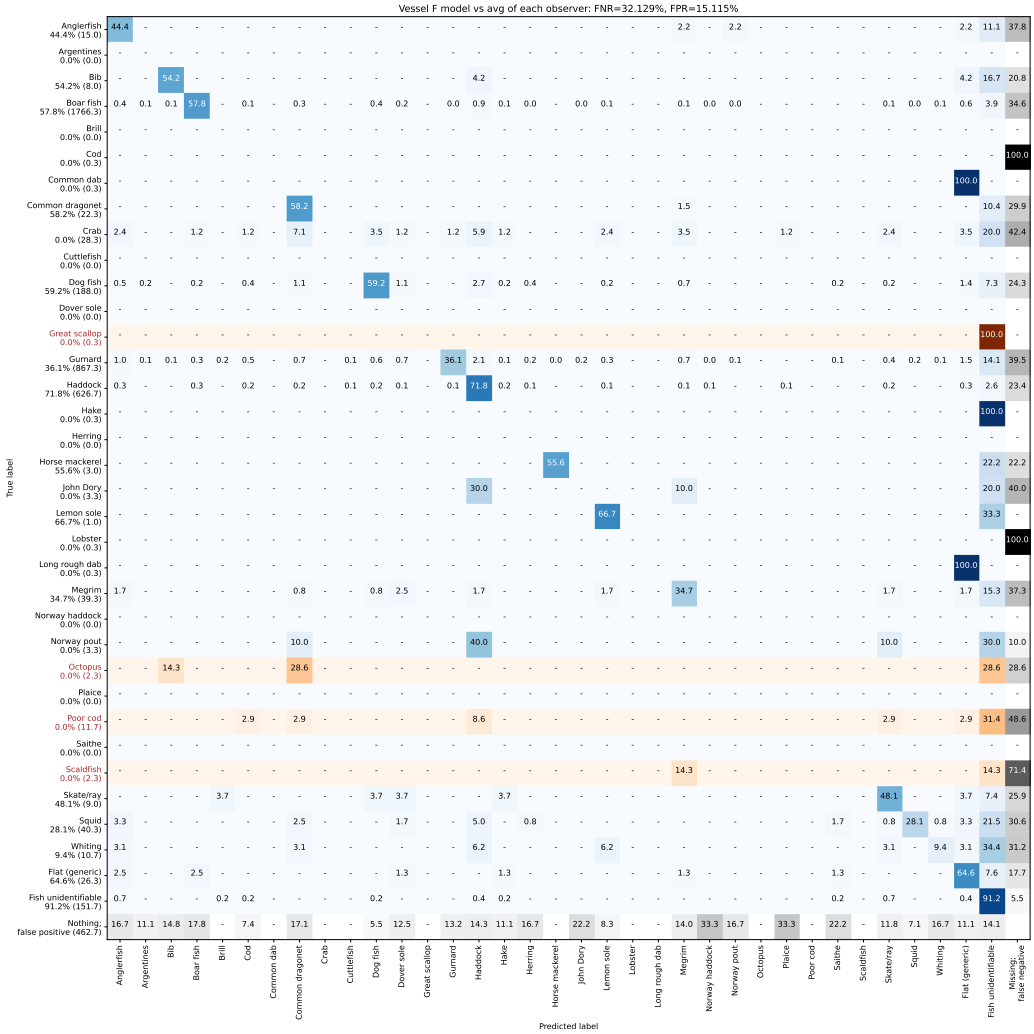


Figure 7.20: Discard counter evaluation for vessel F presented as an extended confusion matrix. The performance of the discard counter is evaluated against the average of multiple observers. Please see the caption of Figure 7.15 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	15.0	36.0	39.32%	17.27%	34.35%	20.73%	41.67%	18.52%	44.44%	26.14%
Argentines	127	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Bib	428	8.0	9.0	69.44%	48.15%	43.06%	45.00%	88.89%	48.15%	54.17%	50.98%
Boar fish	1002	1766.3	1250.0	64.03%	92.50%	58.97%	71.64%	70.77%	81.73%	57.84%	67.74%
Brill	22	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	0.3	9.0	0.00%	0.00%	0.00%	0.00%	3.70%	0.00%	0.00%	0.00%
Common dab	46	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	203	22.3	39.0	53.06%	37.72%	67.03%	44.92%	57.26%	33.33%	58.21%	42.39%
Crab	0	28.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	4	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	188.0	133.0	66.39%	78.42%	60.76%	66.16%	70.74%	83.71%	59.22%	69.37%
Dover sole	360	0.0	16.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Great scallop	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	6040	867.3	362.0	35.75%	94.09%	33.18%	46.67%	41.74%	86.56%	36.13%	50.98%
Haddock	5498	626.7	582.0	83.56%	83.59%	79.07%	80.33%	92.87%	77.26%	71.76%	74.41%
Hake	578	0.3	6.0	3.33%	0.00%	0.00%	0.00%	5.56%	0.00%	0.00%	0.00%
Herring	410	0.0	4.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Horse mackerel	100	3.0	2.0	27.78%	83.33%	33.33%	30.30%	66.67%	83.33%	55.56%	66.67%
John Dory	15	3.3	3.0	13.33%	0.00%	0.00%	0.00%	90.00%	0.00%	0.00%	0.00%
Lemon sole	162	1.0	8.0	4.63%	5.56%	66.67%	8.10%	12.50%	8.33%	66.67%	14.81%
Lobster	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	34	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Megrim	108	39.3	31.0	65.30%	42.36%	32.50%	34.79%	78.81%	44.09%	34.75%	38.86%
Norway haddock	4	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway pout	239	3.3	2.0	15.00%	0.00%	0.00%	0.00%	60.00%	0.00%	0.00%	0.00%
Octopus	0	2.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	534	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Poor cod	0	11.7	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Scaldfish	0	2.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	431	9.0	17.0	36.61%	31.27%	40.41%	26.67%	52.94%	25.49%	48.15%	33.33%
Squid	74	40.3	14.0	37.68%	83.33%	33.36%	45.02%	34.71%	80.95%	28.10%	41.72%
Whiting	446	10.7	4.0	25.19%	27.78%	7.16%	7.96%	37.50%	25.00%	9.37%	13.64%
Flat (generic)	538	26.3	57.0	55.05%	32.45%	62.68%	41.29%	46.20%	29.82%	64.56%	40.80%
Fish unidentifiable	7767	151.7	465.0	33.27%	30.17%	91.10%	43.49%	32.62%	29.75%	91.21%	44.86%
Any species	–	3828.3	3061.0	77.16%	75.39%	58.17%	65.48%	79.96%	68.87%	55.06%	61.20%
Average	–	3828.3	3061.0	20.82%	29.18%	27.54%	17.52%	28.15%	28.00%	28.89%	19.33%

Table 7.15: Discard counter performance species summary for vessel F. The performance of the discard counter is evaluated against the average of multiple observers. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	37.5	98.0	31.29%	10.99%	24.04%	11.62%	38.27%	12.59%	32.89%	18.20%
Argentines	127	146.2	332.0	38.72%	19.45%	49.66%	24.42%	44.03%	24.90%	56.56%	34.58%
Bib	428	707.0	699.0	43.34%	24.99%	46.99%	24.83%	98.87%	75.97%	75.11%	75.53%
Blue whiting	0	56.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Boar fish	1002	1767.7	1259.0	32.01%	52.86%	47.18%	35.82%	71.22%	81.15%	57.80%	67.51%
Brill	22	0.0	25.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Catfish	2	0.7	1.0	66.67%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Cod	447	2.3	136.0	1.39%	0.53%	16.67%	0.81%	1.72%	0.49%	28.57%	0.96%
Common dab	46	38.8	51.0	30.62%	15.00%	13.71%	10.28%	76.14%	11.76%	15.45%	13.36%
Common dragonet	203	162.7	265.0	32.60%	16.65%	49.47%	19.31%	61.38%	18.99%	30.94%	23.54%
Conger eel	1	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Crab	0	412.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	4	5.0	3.0	14.58%	0.00%	0.00%	0.00%	60.00%	0.00%	0.00%	0.00%
Dog fish	376	299.7	316.0	32.17%	25.00%	40.88%	21.94%	94.83%	48.63%	51.28%	49.92%
Dover sole	360	25.0	107.0	14.56%	3.49%	16.00%	3.38%	23.36%	3.12%	13.33%	5.05%
Great scallop	0	13.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	6040	3724.2	3170.0	60.40%	66.28%	42.29%	48.06%	85.12%	66.38%	56.50%	61.05%
Haddock	5498	802.2	1357.0	43.65%	36.21%	61.92%	37.56%	59.11%	39.76%	67.26%	49.97%
Hake	578	634.7	1030.0	14.58%	8.01%	52.40%	10.37%	61.62%	42.94%	69.70%	53.14%
Herring	410	523.2	638.0	34.82%	29.13%	58.73%	29.93%	82.00%	55.98%	68.27%	61.52%
Horse mackerel	100	75.3	73.0	26.06%	25.12%	19.29%	15.44%	96.90%	43.84%	42.48%	43.15%
John Dory	15	3.3	3.0	13.33%	0.00%	0.00%	0.00%	90.00%	0.00%	0.00%	0.00%
Lemon sole	162	16.3	88.0	14.81%	5.88%	49.01%	7.63%	18.56%	5.30%	28.57%	8.95%
Lesser weever fish	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Ling	18	15.2	60.0	28.61%	3.02%	11.61%	4.29%	25.28%	4.44%	17.58%	7.10%
Lobster	0	6.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	34	17.7	124.0	14.56%	2.29%	12.71%	2.57%	14.25%	6.18%	43.40%	10.82%
Mackerel	32	16.3	9.0	33.64%	19.44%	15.28%	12.31%	55.10%	29.63%	16.33%	21.05%
Megrim	108	39.7	44.0	37.32%	26.07%	28.89%	19.88%	90.15%	31.06%	34.45%	32.67%
Norway haddock	4	5.0	51.0	16.00%	10.00%	40.00%	8.89%	9.80%	3.92%	40.00%	7.14%
Norway pout	239	118.3	1790.0	7.43%	2.54%	54.42%	3.95%	6.61%	5.19%	78.45%	9.73%
Octopus	0	15.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	534	835.3	763.0	39.28%	42.11%	56.04%	34.06%	91.34%	64.79%	59.18%	61.86%
Poor cod	0	33.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Queen scallop	0	0.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red mullet	4	0.0	5.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red Stone Crab	0	2.7	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	941.5	1642.0	30.72%	19.92%	56.10%	25.05%	57.34%	50.29%	87.71%	63.93%
Scaldfish	0	19.3	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Shark	30	0.0	27.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	225.3	304.0	50.89%	36.08%	49.76%	35.02%	74.12%	48.03%	64.79%	55.16%
Sole (generic)	0	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Squid	74	45.7	16.0	15.86%	64.81%	15.70%	18.95%	35.04%	70.83%	24.82%	36.76%
Tuskfish	0	3.2	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	446.2	360.0	38.24%	26.99%	21.88%	18.65%	80.69%	43.56%	35.15%	38.91%
Witch	2	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Flat (generic)	538	419.0	565.0	45.36%	27.45%	42.89%	26.90%	74.16%	31.24%	42.12%	35.87%
Fish unidentifiable	7767	371.7	3122.0	16.94%	12.53%	55.89%	14.63%	11.90%	7.38%	62.02%	13.20%
Any species	–	13032.7	18537.0	60.90%	46.79%	53.55%	46.20%	70.31%	40.48%	57.58%	47.54%
Average	–	13032.7	18537.0	19.18%	17.58%	24.40%	10.97%	36.57%	25.79%	30.25%	20.01%

Table 7.16: Discard counter performance species summary for all vessels. The performance of the discard counter is evaluated against the average of multiple observers. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel A	Video 0	Good	Easy	1047.0	1635.0	64.06%	50.03%	78.10%	60.99%
	Video 1	Good	Hard	684.0	1221.0	56.02%	41.52%	74.12%	53.23%
	Video 2	Good	Moderate	242.0	1054.0	22.96%	17.77%	77.41%	28.91%
	Video 3	Poor	Moderate	258.0	453.0	56.95%	5.37%	9.43%	6.84%
	Video 4	Fair	Moderate	357.0	615.0	58.10%	27.43%	47.20%	34.69%
	Average			517.6	995.6	51.62%	28.42%	57.25%	36.93%
Vessel B	Video 0	Good	Hard	204.0	420.0	48.57%	32.06%	66.01%	43.16%
	Video 1	Poor	Moderate	86.0	143.0	60.14%	36.13%	60.08%	45.12%
	Video 2	Good	Moderate	453.0	872.0	51.99%	33.22%	63.90%	43.71%
	Video 3	Good	Easy	408.0	671.0	60.80%	47.24%	77.70%	58.76%
	Video 4	Fair	Moderate	627.0	1022.0	61.42%	48.43%	78.86%	60.01%
	Average			355.6	625.6	56.58%	39.42%	69.31%	50.15%
Vessel C	Video 0	Poor	Moderate	162.0	320.0	50.62%	26.35%	52.06%	34.99%
	Video 1	Good	Hard	198.0	435.0	45.67%	28.81%	63.09%	39.56%
	Video 2	Good	Easy	4.0	20.0	22.50%	10.00%	44.44%	16.33%
	Video 3	Good	Moderate	269.0	873.0	30.85%	17.89%	57.97%	27.34%
	Video 4	Fair	Moderate	515.0	855.0	60.31%	33.53%	55.59%	41.83%
	Average			229.6	500.6	41.99%	23.32%	54.63%	32.01%
Vessel D	Video 0	Good	Hard	199.0	629.0	31.64%	17.57%	55.53%	26.69%
	Video 1	Fair	Moderate	138.0	484.0	28.65%	16.18%	56.49%	25.16%
	Video 2	Poor	Moderate	289.0	655.0	44.22%	22.60%	51.09%	31.33%
	Video 3	Good	Easy	266.0	615.0	43.33%	29.92%	69.04%	41.75%
	Video 4	Good	Moderate	97.0	388.0	25.00%	14.35%	57.39%	22.96%
	Average			197.8	554.2	34.57%	20.12%	57.91%	29.58%
Vessel E	Video 0	Poor	Moderate	99.0	79.0	79.26%	52.74%	41.81%	46.64%
	Video 1	Poor	Moderate	401.0	376.0	93.77%	76.77%	71.99%	74.30%
	Video 2	Moderate	Easy	153.0	95.0	61.96%	43.51%	26.96%	33.29%
	Video 3	Poor	Moderate	64.0	40.0	62.50%	63.33%	39.58%	48.72%
	Video 4	Poor	Moderate	86.0	60.0	69.77%	40.00%	27.91%	32.88%
	Video 5	Moderate	Moderate	484.0	409.0	84.71%	76.02%	64.39%	69.72%
	Video 6	Moderate	Moderate-easy	271.0	226.0	83.39%	64.01%	53.38%	58.22%
	Video 7	Good	Moderate	467.0	341.0	72.97%	0.00%	0.00%	0.00%
	Video 8	Good	Moderate	33.0	13.0	39.00%	74.36%	29.00%	41.73%
	Video 9	Good	Moderate	166.0	113.0	67.94%	54.28%	36.87%	43.91%
	Video 10	Poor	Moderate	161.0	109.0	68.32%	60.30%	41.20%	48.95%
	Video 11	Good	Moderate-easy	12.0	10.0	81.08%	56.67%	45.95%	50.75%
	Video 12	Good	Moderate-easy	296.0	222.0	75.00%	68.32%	51.24%	58.56%
	Average			207.153846	161.0	72.28%	56.18%	40.79%	46.74%
Vessel F	Video 0	Fair	Moderate-easy	1415.0	1187.0	83.85%	55.43%	46.48%	50.56%
	Video 1	Fair	Hard	695.0	576.0	82.84%	80.09%	66.35%	72.57%
	Video 2	Fair	Moderate	318.0	263.0	82.70%	80.61%	66.67%	72.98%
	Video 3	Good	Moderate	304.0	193.0	63.35%	73.06%	46.28%	56.66%
	Video 4	Good	Moderate-easy	225.0	178.0	79.11%	76.03%	60.15%	67.16%
	Video 5	Good	Hard	162.0	130.0	80.25%	71.28%	57.20%	63.47%
	Video 6	Good	Moderate	390.0	327.0	83.77%	74.62%	62.51%	68.03%
	Video 7	Good	Easy	79.0	62.0	77.50%	95.16%	73.75%	83.10%
	Video 8	Moderate	Hard	237.0	145.0	61.10%	72.18%	44.10%	54.75%
	Average			425.0	340.111111	77.16%	75.39%	58.17%	65.48%

Table 7.17: Per-video discard counter performance. The performance of the discard counter is evaluated against the average of multiple observers.

Inter-observer results

We now evaluate the performance of human expert observers in comparison to one another. We present our results using extended confusion matrices, species summary tables and video summary tables. For a given vessel or video, we evaluate the performance of each observer in comparison to the other human observers. We then compute the average of the resulting metrics (*e.g.* confusion matrices or summary tables) to summaries the expected average performance of a human observer.

We note that the sets of observers that quantified the Marine Scotland Science (MSS) footage is distinct from those that quantified the Cefas footage. We refer to the three observers that quantified the MSS footage as U, V and W, while we refer to the Cefas observers as X, Y and Z.

We present our inter-observer performance as extended confusion matrices for vessels A-F in Figures 7.21, 7.22, 7.23, 7.24, 7.25 and 7.26.

The per-species inter-observer performance is summarised for vessel A-F in Tables 7.18, 7.19, 7.20, 7.21, 7.22 and 7.23, for all MSS vessels in Table 7.24 and for all Cefas vessels in Table 7.25.

Per-video inter-observer performance for MSS and Cefas vessels is summarised in Tables 7.26 and 7.27.

Figure 7.21: Inter-observer performance evaluation for vessel A presented as an extended confusion matrix. Orange rows indicate species that were not present in the training set, hence we could not expect equivalent performance from our system. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from different assessments from observers.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	2.3	22.22%	41.67%	50.00%	29.63%	75.56%	41.67%	45.00%	42.22%
Argentines	19.3	21.87%	17.65%	13.25%	7.80%	52.37%	18.17%	12.16%	12.44%
Blue whiting	56.3	37.60%	28.99%	22.94%	16.34%	74.93%	33.58%	33.16%	32.49%
Catfish	0.7	33.33%	–	66.67%	44.44%	33.33%	–	66.67%	44.44%
Cod	1.3	61.11%	50.00%	55.56%	48.89%	61.11%	50.00%	55.56%	48.89%
Common dab	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Crab	0.7	33.33%	–	66.67%	44.44%	33.33%	–	66.67%	44.44%
Dog fish	3.3	33.33%	–	40.00%	26.67%	33.33%	–	40.00%	26.67%
Gurnard	458.3	66.24%	42.10%	44.96%	38.88%	68.99%	46.39%	48.37%	45.44%
Haddock	52.3	50.39%	30.72%	25.91%	22.87%	96.88%	38.11%	38.29%	38.19%
Hake	625.7	75.54%	40.04%	40.05%	38.24%	92.41%	43.78%	43.86%	43.72%
Herring	58.3	56.05%	27.12%	23.06%	21.38%	70.69%	36.11%	39.65%	36.47%
Horse mackerel	59.3	50.96%	22.24%	19.32%	15.14%	75.85%	36.29%	36.79%	35.76%
Lemon sole	1.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Ling	14.7	64.96%	41.50%	42.16%	37.75%	72.55%	41.47%	42.16%	40.62%
Long rough dab	1.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Mackerel	12.7	35.28%	35.71%	27.61%	21.82%	87.78%	48.27%	47.27%	47.50%
Norway haddock	5.0	69.05%	39.68%	43.06%	39.23%	69.05%	39.68%	43.06%	39.23%
Norway pout	27.0	20.90%	9.31%	16.36%	5.99%	47.13%	11.41%	20.31%	13.51%
Plaice	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red Stone Crab	1.3	33.33%	–	66.67%	44.44%	33.33%	–	66.67%	44.44%
Saithe	838.3	61.95%	43.41%	50.17%	42.70%	79.58%	51.84%	52.62%	51.46%
Skate/ray	45.7	46.77%	45.97%	58.15%	43.23%	74.32%	41.75%	43.55%	41.15%
Squid	0.7	33.33%	–	0.00%	0.00%	33.33%	–	0.00%	0.00%
Tuskfish	2.7	61.11%	12.50%	13.89%	12.22%	61.11%	12.50%	13.89%	12.22%
Whiting	225.3	62.40%	39.73%	45.08%	39.32%	70.71%	43.23%	45.17%	42.82%
Flat (generic)	31.7	35.15%	32.97%	33.94%	22.52%	57.06%	36.59%	40.66%	34.91%
Fish unidentifiable	43.0	51.28%	7.18%	6.57%	5.57%	59.10%	8.80%	8.00%	7.84%
Any species	2588.7	77.71%	40.68%	42.89%	40.36%	81.93%	44.56%	45.39%	44.42%
Average	2588.7	39.91%	30.98%	33.00%	23.91%	54.07%	34.06%	35.97%	29.53%

Table 7.18: Inter-observer variability species summary table for vessel A, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

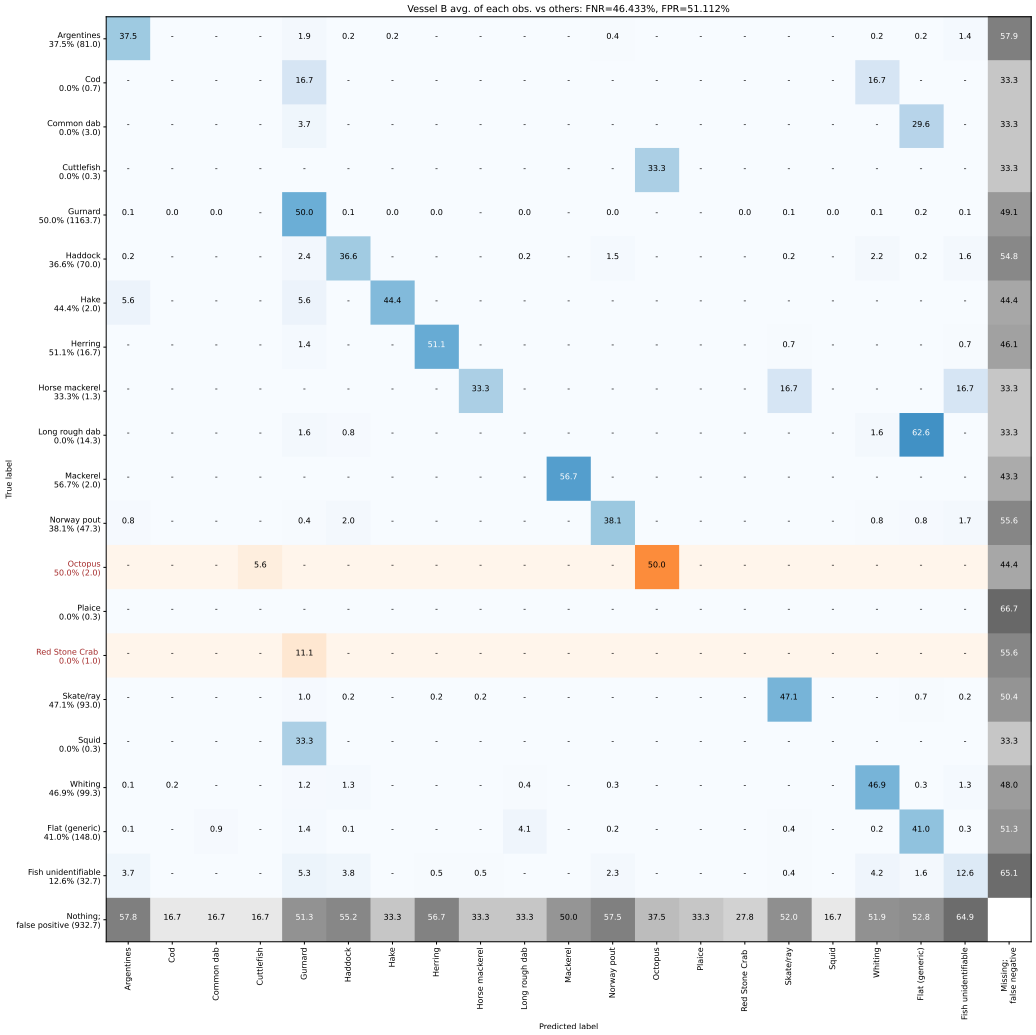


Figure 7.22: Inter-observer discard count evaluation for vessel B presented as an extended confusion matrix. Please see the caption of Figure 7.21 for a more detailed description.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	81.0	66.86%	40.10%	46.16%	38.23%	91.81%	37.60%	37.49%	37.46%
Cod	0.7	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	3.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	1163.7	69.25%	46.19%	50.46%	45.42%	72.30%	47.67%	49.95%	47.17%
Haddock	70.0	71.58%	31.97%	35.24%	32.22%	86.93%	36.01%	36.62%	36.06%
Hake	2.0	25.00%	–	47.22%	24.81%	33.33%	–	44.44%	29.63%
Herring	16.7	50.27%	42.14%	49.70%	40.26%	46.18%	41.15%	51.06%	39.34%
Horse mackerel	1.3	20.83%	–	33.33%	15.00%	33.33%	–	33.33%	22.22%
Long rough dab	14.3	1.30%	–	0.00%	0.00%	4.07%	–	0.00%	0.00%
Mackerel	2.0	33.33%	50.00%	66.67%	44.44%	63.33%	50.00%	56.67%	48.41%
Norway pout	47.3	77.58%	36.08%	38.06%	36.19%	77.58%	36.08%	38.06%	36.19%
Octopus	2.0	18.06%	16.67%	44.44%	17.86%	41.67%	16.67%	50.00%	30.00%
Plaice	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red Stone Crab	1.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	93.0	56.11%	45.54%	53.63%	43.53%	74.43%	45.66%	47.13%	45.18%
Squid	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	99.3	49.96%	36.63%	44.23%	33.22%	61.94%	43.55%	46.94%	41.58%
Flat (generic)	148.0	59.06%	29.80%	32.45%	28.89%	68.76%	40.47%	40.95%	38.97%
Fish unidentifiable	32.7	60.13%	11.28%	10.96%	9.78%	63.56%	13.30%	12.63%	12.30%
Any species	1779.0	70.07%	43.52%	47.30%	42.71%	74.34%	44.05%	45.77%	43.68%
Average	1779.0	32.97%	30.47%	31.99%	20.49%	40.96%	32.11%	31.56%	23.23%

Table 7.19: Inter-observer variability species summary table for vessel B, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	32.7	64.32%	28.06%	29.18%	27.03%	81.76%	28.47%	29.18%	28.40%
Common dab	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	673.7	75.47%	57.32%	58.41%	55.37%	70.95%	47.62%	50.08%	46.56%
Haddock	40.7	38.57%	41.17%	40.24%	31.18%	77.55%	26.38%	24.35%	24.80%
Hake	5.7	44.44%	29.17%	30.56%	22.75%	63.59%	35.42%	38.43%	34.49%
Herring	140.7	53.45%	46.83%	50.80%	40.04%	69.42%	55.25%	56.42%	53.68%
Horse mackerel	11.0	24.74%	25.00%	17.55%	12.64%	66.40%	32.08%	32.20%	29.95%
Mackerel	0.7	33.33%	–	66.67%	44.44%	33.33%	–	66.67%	44.44%
Norway pout	32.3	18.34%	14.85%	29.71%	11.24%	38.93%	11.18%	22.35%	13.87%
Red Stone Crab	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	97.3	42.91%	41.90%	41.28%	30.51%	47.07%	44.46%	50.53%	40.19%
Skate/ray	10.7	42.20%	47.56%	48.92%	35.92%	71.91%	55.09%	55.12%	52.83%
Whiting	48.7	69.55%	44.63%	44.48%	41.67%	66.21%	40.60%	42.33%	38.40%
Flat (generic)	7.3	37.22%	35.37%	36.67%	23.31%	49.72%	36.81%	38.57%	30.39%
Fish unidentifiable	47.0	26.18%	6.94%	4.40%	4.13%	9.52%	6.80%	2.98%	1.59%
Any species	1149.0	66.70%	47.87%	49.01%	45.70%	74.70%	42.56%	44.75%	42.45%
Average	1149.0	38.05%	35.18%	35.70%	25.35%	49.76%	35.24%	36.56%	29.31%

Table 7.20: Inter-observer variability species summary table for vessel C, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

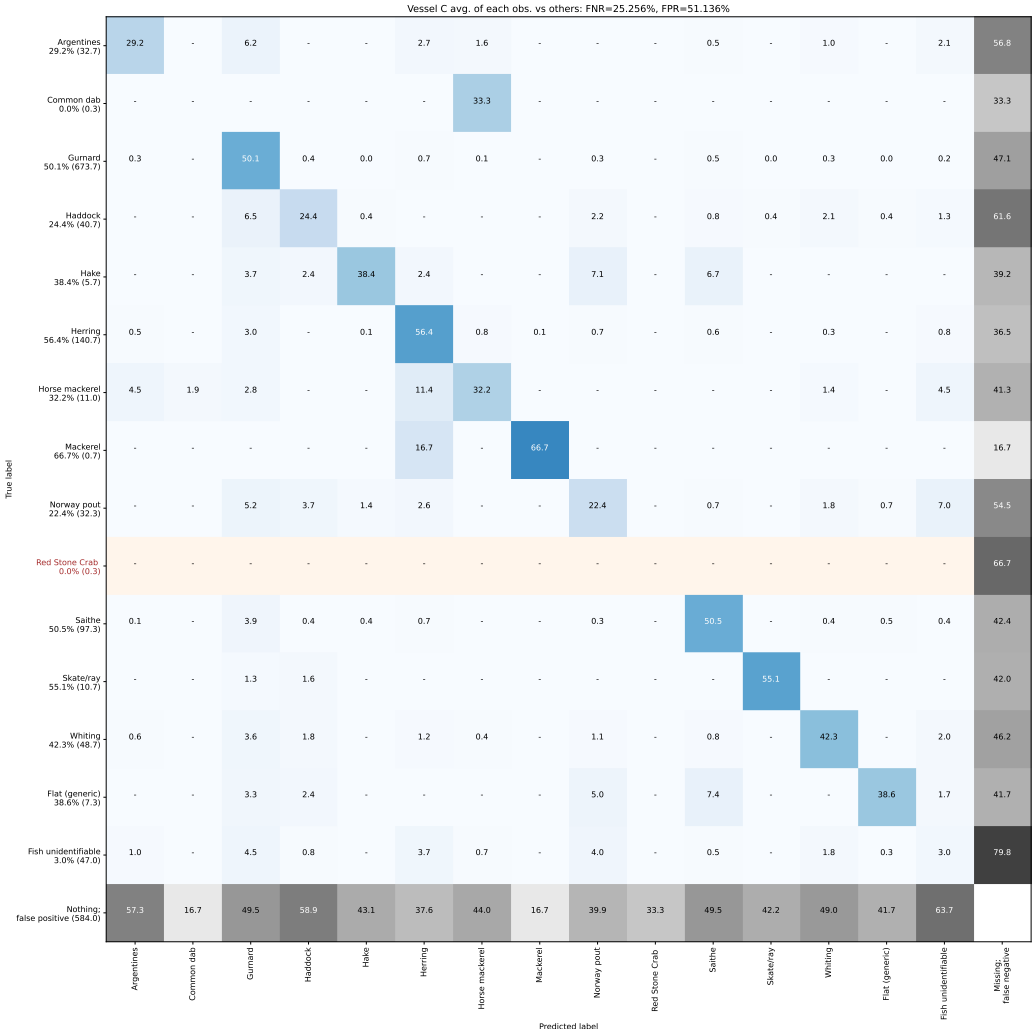


Figure 7.23: Inter-observer discard count evaluation for vessel C presented as an extended confusion matrix. Please see the caption of Figure 7.21 for a more detailed description.

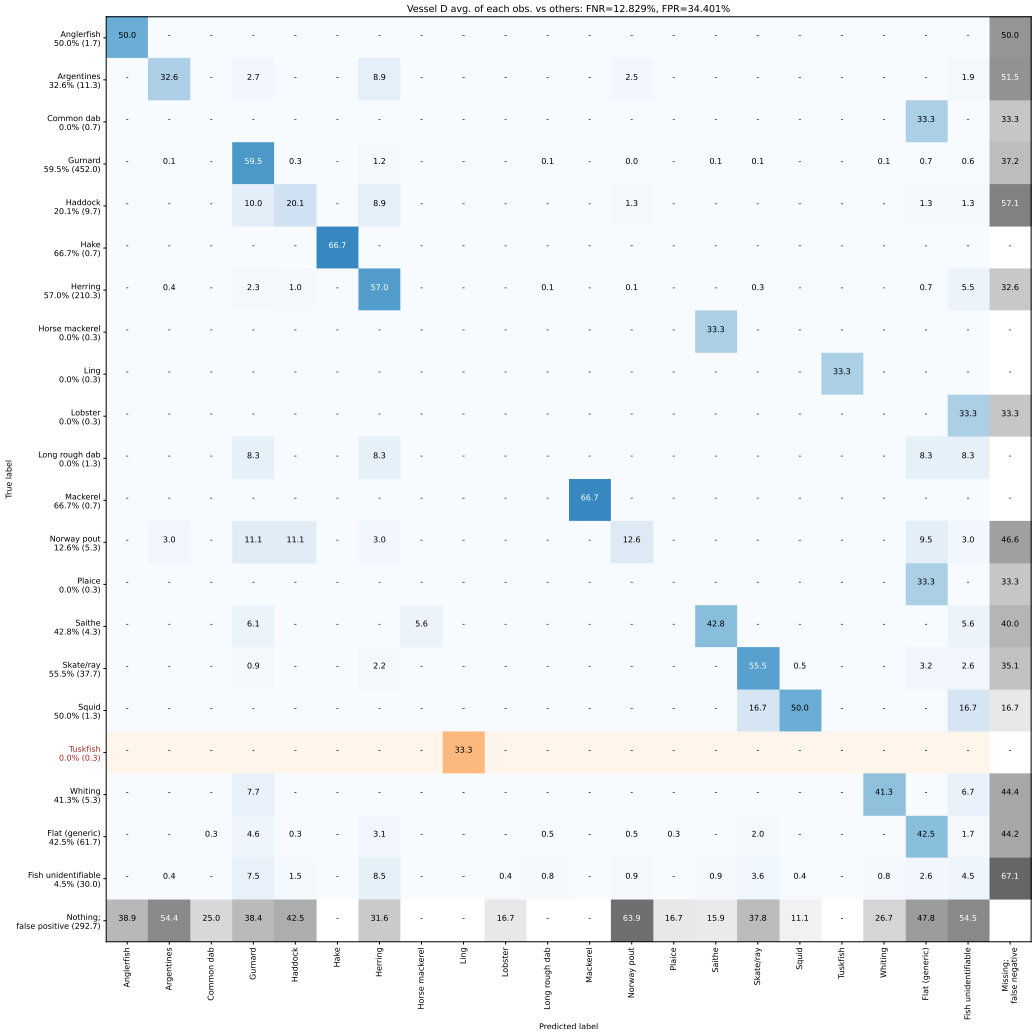


Figure 7.24: Inter-observer discard count evaluation for vessel D presented as an extended confusion matrix. Please see the caption of Figure 7.21 for a more detailed description.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	1.7	22.22%	–	55.56%	29.63%	44.44%	–	50.00%	33.33%
Argentines	11.3	36.50%	27.99%	37.38%	24.04%	76.22%	29.84%	32.61%	30.76%
Common dab	0.7	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	452.0	60.78%	55.55%	53.26%	48.69%	52.05%	58.48%	59.55%	51.77%
Haddock	9.7	25.53%	10.32%	7.98%	7.11%	63.31%	25.45%	20.13%	22.46%
Hake	0.7	–	–	–	–	–	–	–	–
Herring	210.3	54.45%	48.31%	51.56%	44.17%	72.27%	63.83%	57.04%	57.86%
Horse mackerel	0.3	–	–	–	–	–	–	–	–
Ling	0.3	–	–	–	–	–	–	–	–
Lobster	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	1.3	–	–	–	–	–	–	–	–
Mackerel	0.7	–	–	–	–	–	–	–	–
Norway pout	5.3	22.45%	9.57%	6.79%	3.03%	41.04%	13.89%	12.55%	9.06%
Plaice	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	4.3	27.08%	–	43.06%	29.72%	57.14%	–	42.78%	38.89%
Skate/ray	37.7	54.75%	50.67%	49.94%	44.87%	57.20%	49.51%	55.54%	48.66%
Squid	1.3	33.33%	–	50.00%	33.33%	27.78%	–	50.00%	38.89%
Tuskfish	0.3	–	–	–	–	–	–	–	–
Whiting	5.3	41.67%	–	46.67%	34.89%	49.47%	–	41.28%	33.95%
Flat (generic)	61.7	55.61%	46.10%	46.15%	40.34%	69.30%	40.81%	42.54%	40.09%
Fish unidentifiable	30.0	32.15%	5.94%	7.98%	5.18%	39.22%	6.44%	4.48%	4.51%
Any species	835.7	62.63%	52.97%	50.09%	46.17%	64.94%	52.81%	53.79%	50.56%
Average	835.7	29.98%	34.29%	33.92%	23.17%	39.22%	36.05%	34.66%	26.41%

Table 7.21: Inter-observer variability species summary table for vessel D, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

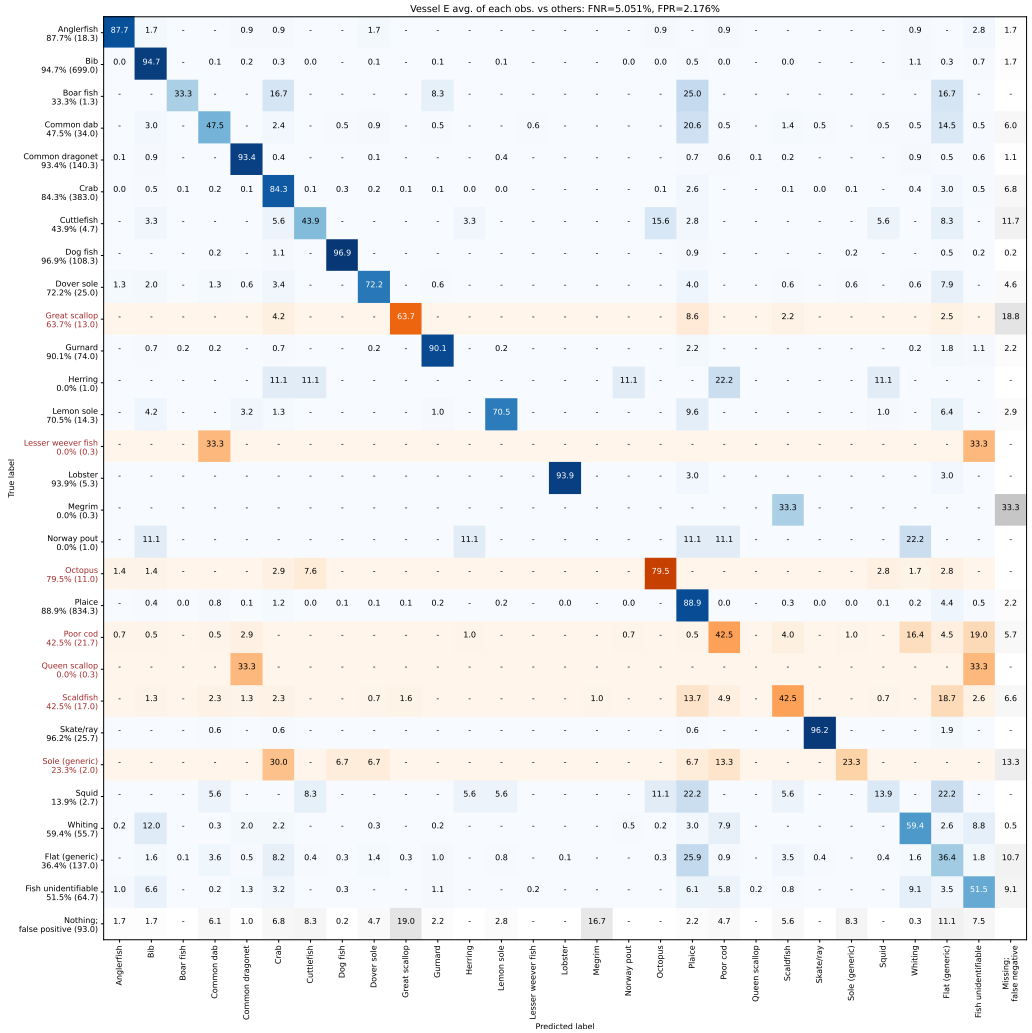


Figure 7.25: Inter-observer discard count evaluation for vessel E presented as an extended confusion matrix. Please see the caption of Figure 7.21 for a more detailed description.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	18.3	79.49%	89.20%	88.52%	85.11%	86.77%	88.11%	87.68%	87.28%
Bib	699.0	90.89%	94.57%	93.90%	93.51%	95.53%	94.65%	94.67%	94.60%
Boar fish	1.3	20.83%	8.33%	33.33%	15.00%	33.33%	8.33%	33.33%	22.22%
Common dab	34.0	52.37%	49.29%	51.18%	42.28%	79.40%	48.18%	47.45%	47.12%
Common dragonet	140.3	89.73%	93.27%	92.46%	91.98%	96.75%	93.38%	93.38%	93.34%
Crab	383.0	82.80%	83.78%	82.64%	81.13%	89.22%	84.30%	84.29%	83.95%
Cuttlefish	4.7	16.67%	35.00%	42.78%	22.22%	50.28%	35.42%	43.89%	34.13%
Dog fish	108.3	93.00%	91.15%	88.37%	88.37%	97.58%	96.95%	96.93%	96.92%
Dover sole	25.0	52.59%	69.69%	66.84%	54.77%	92.45%	72.09%	72.18%	71.97%
Great scallop	13.0	44.52%	59.31%	46.53%	40.28%	67.83%	64.72%	63.68%	61.25%
Gurnard	74.0	78.51%	81.97%	81.19%	77.40%	98.66%	90.09%	90.10%	90.09%
Herring	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Lemon sole	14.3	61.93%	73.99%	71.07%	63.70%	63.88%	72.26%	70.49%	67.00%
Lesser weever fish	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	5.3	83.33%	94.44%	88.89%	83.33%	88.38%	94.44%	93.94%	93.80%
Megrim	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Norway pout	1.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Octopus	11.0	58.10%	80.00%	70.44%	62.60%	83.97%	79.98%	79.49%	78.76%
Plaice	834.3	84.55%	87.43%	86.87%	85.58%	92.34%	88.94%	88.90%	88.72%
Poor cod	21.7	17.14%	–	30.62%	11.39%	29.51%	–	42.48%	22.45%
Queen scallop	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Scaldfish	17.0	20.49%	5.26%	41.07%	17.95%	38.10%	6.67%	42.50%	24.78%
Skate/ray	25.7	94.23%	96.28%	95.86%	95.15%	94.97%	96.20%	96.15%	96.10%
Sole (generic)	2.0	10.42%	19.44%	20.83%	7.50%	35.00%	20.83%	23.33%	16.19%
Squid	2.7	13.33%	13.89%	15.00%	8.89%	61.11%	12.50%	13.89%	12.22%
Whiting	55.7	34.12%	71.23%	60.51%	45.43%	38.97%	69.71%	59.37%	50.35%
Flat (generic)	137.0	52.28%	40.54%	38.47%	32.46%	70.50%	36.49%	36.42%	35.19%
Fish unidentifiable	64.7	41.89%	42.00%	35.28%	26.81%	55.95%	57.51%	51.52%	46.87%
Any species	2695.3	93.40%	78.52%	78.56%	78.34%	96.21%	83.54%	83.55%	83.51%
Average	2695.3	45.47%	59.99%	54.00%	44.03%	58.59%	61.55%	57.18%	50.55%

Table 7.22: Inter-observer variability species summary table for vessel E, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

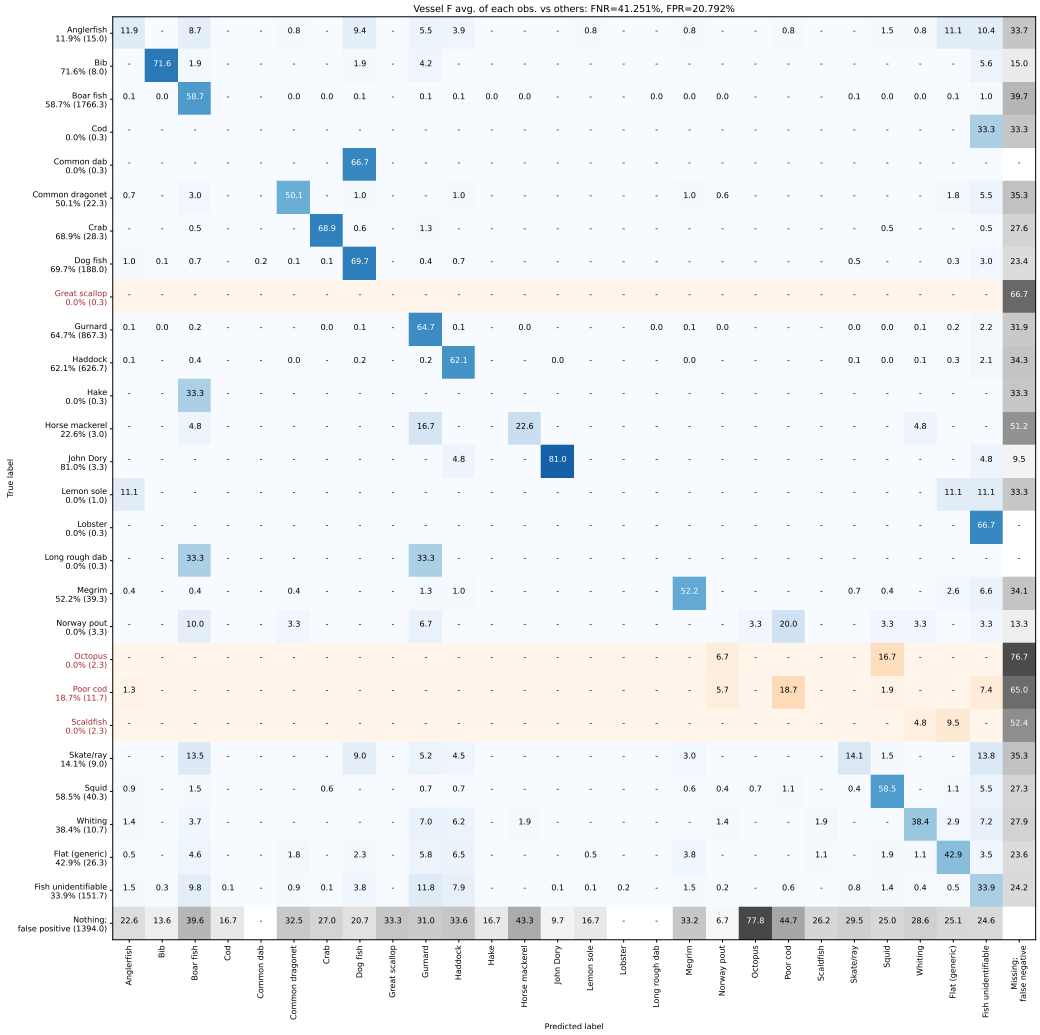


Figure 7.26: Inter-observer discard count evaluation for vessel F presented as an extended confusion matrix. Please see the caption of Figure 7.21 for a more detailed description.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	15.0	4.25%	–	11.46%	0.93%	5.95%	–	11.90%	2.16%
Bib	8.0	71.69%	65.28%	58.92%	58.78%	78.89%	73.89%	71.56%	71.01%
Boar fish	1766.3	68.24%	72.95%	71.30%	68.29%	72.00%	58.79%	58.67%	56.86%
Cod	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	22.3	57.06%	65.80%	64.10%	54.46%	61.86%	52.67%	50.06%	46.00%
Crab	28.3	75.08%	79.49%	79.16%	75.23%	79.92%	69.56%	68.93%	68.20%
Dog fish	188.0	69.69%	73.29%	71.72%	68.90%	63.37%	72.29%	69.65%	66.76%
Great scallop	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	867.3	67.61%	75.41%	73.68%	70.61%	70.59%	65.75%	64.75%	62.55%
Haddock	626.7	77.02%	76.51%	75.19%	73.83%	80.49%	62.73%	62.09%	61.26%
Hake	0.3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Horse mackerel	3.0	36.11%	27.78%	14.81%	13.76%	51.43%	28.33%	22.62%	22.94%
John Dory	3.3	66.67%	81.94%	83.33%	74.60%	82.14%	80.56%	80.95%	79.85%
Lemon sole	1.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	0.3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Megrim	39.3	49.08%	64.26%	55.86%	46.74%	55.07%	54.50%	52.20%	48.38%
Norway pout	3.3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Octopus	2.3	75.56%	0.00%	0.00%	0.00%	75.56%	0.00%	0.00%	0.00%
Poor cod	11.7	24.80%	–	37.38%	17.57%	33.33%	–	18.67%	10.37%
Scaldfish	2.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	9.0	31.43%	29.89%	12.70%	11.96%	35.08%	26.76%	14.11%	16.27%
Squid	40.3	72.79%	69.91%	67.67%	64.37%	60.35%	60.33%	58.55%	55.40%
Whiting	10.7	40.28%	47.62%	40.52%	30.18%	73.60%	38.49%	38.41%	37.43%
Flat (generic)	26.3	53.02%	52.40%	48.58%	42.72%	69.98%	43.84%	42.94%	41.65%
Fish unidentifiable	151.7	41.35%	45.84%	40.12%	31.29%	51.30%	39.48%	33.86%	29.34%
Any species	3828.3	75.97%	69.78%	69.17%	67.63%	75.56%	58.96%	58.71%	57.44%
Average	3828.3	36.36%	47.52%	38.23%	29.79%	40.77%	42.05%	34.64%	28.76%

Table 7.23: Inter-observer variability species summary table for vessel F, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	4.0	23.33%	41.67%	53.33%	31.11%	77.96%	42.50%	42.66%	41.74%
Argentines	144.3	48.46%	29.90%	32.67%	25.38%	93.53%	31.88%	31.77%	31.78%
Blue whiting	56.3	37.60%	28.99%	22.94%	16.34%	74.93%	33.58%	33.16%	32.49%
Catfish	0.7	33.33%	–	66.67%	44.44%	33.33%	–	66.67%	44.44%
Cod	2.0	30.56%	41.67%	44.44%	24.44%	35.00%	41.67%	46.67%	32.38%
Common dab	4.3	0.00%	–	0.00%	0.00%	26.67%	–	0.00%	0.00%
Crab	0.7	33.33%	–	66.67%	44.44%	33.33%	–	66.67%	44.44%
Cuttlefish	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Dog fish	3.3	33.33%	–	40.00%	26.67%	33.33%	–	40.00%	26.67%
Gurnard	2747.7	68.50%	50.15%	52.04%	47.28%	67.15%	48.74%	50.98%	47.78%
Haddock	172.7	47.86%	30.12%	28.34%	23.83%	93.23%	33.43%	33.58%	33.44%
Hake	634.0	53.59%	40.55%	42.31%	33.47%	91.93%	43.74%	43.82%	43.66%
Herring	426.0	54.77%	40.06%	42.54%	35.86%	78.01%	59.97%	58.46%	58.17%
Horse mackerel	72.0	34.79%	22.61%	20.20%	13.16%	74.15%	35.40%	35.49%	34.53%
Lemon sole	1.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Ling	15.0	57.28%	39.72%	39.13%	33.32%	73.04%	40.87%	40.79%	39.68%
Lobster	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	16.7	0.74%	–	0.00%	0.00%	3.26%	–	0.00%	0.00%
Mackerel	16.0	39.58%	45.95%	49.36%	36.66%	95.95%	49.66%	49.77%	49.69%
Norway haddock	5.0	69.05%	39.68%	43.06%	39.23%	69.05%	39.68%	43.06%	39.23%
Norway pout	112.0	23.98%	15.97%	20.44%	9.56%	58.87%	24.89%	26.19%	23.66%
Octopus	2.0	18.06%	16.67%	44.44%	17.86%	41.67%	16.67%	50.00%	30.00%
Plaice	1.0	0.00%	–	0.00%	0.00%	41.67%	–	0.00%	0.00%
Red Stone Crab	2.7	8.33%	–	27.78%	11.11%	37.78%	–	35.56%	22.38%
Saithe	940.0	47.11%	44.20%	46.25%	36.04%	75.47%	51.03%	52.08%	50.39%
Skate/ray	187.0	50.72%	47.92%	53.62%	42.85%	75.63%	46.34%	47.76%	46.03%
Squid	2.3	25.00%	–	22.22%	16.67%	44.44%	–	30.00%	23.23%
Tuskfish	3.0	48.15%	11.11%	13.89%	10.37%	52.22%	11.67%	13.89%	11.43%
Whiting	378.7	56.86%	41.96%	45.83%	38.06%	66.66%	42.84%	44.83%	41.93%
Flat (generic)	248.7	48.11%	36.14%	37.48%	29.80%	66.93%	40.19%	41.33%	38.73%
Fish unidentifiable	152.7	43.91%	8.09%	7.70%	6.37%	37.47%	9.11%	6.90%	6.34%
Any species	6352.3	69.91%	46.14%	47.64%	44.05%	77.23%	45.40%	46.69%	45.20%
Average	6352.3	33.43%	31.37%	32.42%	22.40%	53.31%	34.24%	34.69%	28.85%

Table 7.24: Inter-observer variability species summary table for all MSS vessels, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Species	Avg. count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	33.3	41.87%	67.92%	53.99%	43.02%	47.76%	62.44%	53.32%	49.86%
Bib	707.0	86.09%	87.25%	85.15%	84.82%	95.31%	94.39%	94.40%	94.33%
Boar fish	1767.7	58.76%	68.52%	64.98%	57.63%	71.99%	58.76%	58.64%	56.84%
Cod	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	34.3	48.34%	48.25%	48.89%	39.03%	78.09%	47.89%	47.04%	46.67%
Common dragonet	162.7	74.36%	80.74%	79.12%	74.33%	90.56%	87.18%	87.08%	86.88%
Crab	411.3	79.71%	81.98%	81.25%	78.77%	88.55%	83.26%	83.23%	82.86%
Cuttlefish	4.7	16.67%	35.00%	42.78%	22.22%	50.28%	35.42%	43.89%	34.13%
Dog fish	296.3	83.67%	83.88%	81.71%	80.58%	75.21%	80.46%	78.97%	77.80%
Dover sole	25.0	52.59%	69.69%	66.84%	54.77%	92.45%	72.09%	72.18%	71.97%
Great scallop	13.3	38.16%	56.44%	42.03%	34.52%	66.99%	63.62%	62.30%	59.71%
Gurnard	941.3	72.74%	78.22%	77.21%	73.81%	72.44%	67.56%	66.62%	64.73%
Haddock	626.7	77.02%	76.51%	75.19%	73.83%	80.49%	62.73%	62.09%	61.26%
Hake	0.3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Herring	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Horse mackerel	3.0	36.11%	27.78%	14.81%	13.76%	51.43%	28.33%	22.62%	22.94%
John Dory	3.3	66.67%	81.94%	83.33%	74.60%	82.14%	80.56%	80.95%	79.85%
Lemon sole	15.3	35.39%	64.77%	52.92%	36.40%	62.31%	69.49%	66.45%	62.69%
Lesser weever fish	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	5.7	71.43%	88.89%	79.37%	71.43%	88.89%	88.89%	88.38%	88.27%
Long rough dab	0.3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Megrim	39.7	43.63%	63.03%	53.00%	41.55%	54.40%	54.32%	51.91%	47.99%
Norway pout	4.3	0.00%	0.00%	0.00%	0.00%	25.00%	0.00%	0.00%	0.00%
Octopus	13.3	60.28%	67.92%	60.79%	54.78%	84.02%	65.67%	65.47%	64.96%
Plaice	834.3	84.55%	87.43%	86.87%	85.58%	92.34%	88.94%	88.90%	88.72%
Poor cod	33.3	22.50%	–	32.66%	15.71%	35.08%	–	27.37%	17.83%
Queen scallop	0.3	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Scaldfish	19.3	18.44%	5.26%	36.71%	16.16%	36.31%	6.67%	33.62%	21.56%
Skate/ray	34.7	64.92%	72.00%	62.60%	56.33%	78.98%	77.32%	75.40%	75.22%
Sole (generic)	2.0	10.42%	19.44%	20.83%	7.50%	35.00%	20.83%	23.33%	16.19%
Squid	43.0	49.92%	57.31%	51.24%	43.03%	63.31%	56.67%	55.12%	52.75%
Whiting	66.3	38.08%	54.49%	46.19%	35.63%	47.20%	59.26%	52.86%	48.28%
Flat (generic)	163.3	52.56%	44.58%	42.22%	36.37%	72.02%	37.40%	37.35%	36.25%
Fish unidentifiable	216.3	41.65%	43.02%	37.22%	28.80%	62.13%	36.97%	36.75%	34.56%
Any species	6523.7	86.27%	74.94%	74.71%	73.96%	83.44%	68.96%	68.81%	68.22%
Average	6523.7	41.96%	56.62%	48.67%	39.26%	55.31%	55.55%	50.46%	45.44%

Table 7.25: Inter-observer variability species summary table for all Cefas vessels, showing average metrics for the performance of each observer vs. other observers. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Vessel	Video	Quality	Difficulty	Avg. count	Count accuracy	Precision	Recall	F1
Vessel A	Video 0	Good	Easy	1047.3	83.43%	52.54%	53.43%	52.46%
	Video 1	Good	Hard	684.0	91.54%	42.27%	42.37%	42.22%
	Video 2	Good	Moderate	242.0	61.21%	40.73%	46.92%	39.67%
	Video 3	Poor	Moderate	258.0	86.20%	29.41%	29.37%	29.18%
	Video 4	Fair	Moderate	357.3	66.17%	38.45%	42.35%	38.27%
	Average			517.733333	77.71%	40.68%	42.89%	40.36%
Vessel B	Video 0	Good	Hard	204.0	76.23%	41.18%	41.93%	40.60%
	Video 1	Poor	Moderate	86.0	63.09%	45.01%	50.13%	43.68%
	Video 2	Good	Moderate	453.3	86.13%	39.66%	40.30%	39.64%
	Video 3	Good	Easy	408.0	44.57%	44.70%	56.06%	42.89%
	Video 4	Fair	Moderate	627.7	80.35%	47.05%	48.08%	46.75%
	Average			355.8	70.07%	43.52%	47.30%	42.71%
Vessel C	Video 0	Poor	Moderate	162.0	72.55%	51.46%	53.03%	50.65%
	Video 1	Good	Hard	198.7	68.25%	50.97%	50.77%	47.86%
	Video 2	Good	Easy	4.5	80.00%	67.50%	67.50%	66.67%
	Video 3	Good	Moderate	269.7	66.45%	33.81%	32.84%	31.82%
	Video 4	Fair	Moderate	515.7	51.84%	44.03%	51.76%	41.96%
	Average			230.1	67.82%	49.56%	51.18%	47.79%
Vessel D	Video 0	Good	Hard	199.0	94.15%	76.45%	76.45%	76.38%
	Video 1	Fair	Moderate	138.7	49.09%	43.11%	50.23%	40.11%
	Video 2	Poor	Moderate	289.7	45.23%	62.68%	64.70%	52.62%
	Video 3	Good	Easy	266.5	94.53%	78.49%	78.49%	78.42%
	Video 4	Good	Moderate	97.0	67.25%	25.03%	23.60%	23.05%
	Average			198.166667	70.05%	57.15%	58.70%	54.12%

Table 7.26: Per-video inter-observer summary for average of all MSS observers (U, V and W), vessels A-D

Vessel	Video	Quality	Difficulty	Avg. count	Count accuracy	Precision	Recall	F1
Vessel E	Video 0	Poor	Moderate	99.7	96.73%	88.00%	87.98%	87.96%
	Video 1	Poor	Moderate	401.0	96.59%	89.56%	89.55%	89.52%
	Video 2	Moderate	Easy	153.3	85.35%	79.99%	79.80%	79.32%
	Video 3	Poor	Moderate	64.0	94.03%	70.33%	70.44%	70.28%
	Video 4	Poor	Moderate	86.0	95.51%	64.06%	63.97%	63.96%
	Video 5	Moderate	Moderate	484.0	97.76%	87.41%	87.41%	87.40%
	Video 6	Moderate	Moderate-easy	271.0	96.40%	80.83%	80.84%	80.81%
	Video 7	Good	Moderate	467.3	94.52%	85.44%	85.45%	85.37%
	Video 8	Good	Moderate	33.3	75.25%	55.54%	56.54%	54.73%
	Video 9	Good	Moderate	166.3	98.02%	81.57%	81.58%	81.56%
	Video 10	Poor	Moderate	161.0	96.95%	80.38%	80.34%	80.33%
	Video 11	Good	Moderate-easy	12.3	89.74%	73.43%	73.08%	73.00%
	Video 12	Good	Moderate-easy	296.0	97.35%	84.25%	84.26%	84.23%
	Average			207.333333	93.40%	78.52%	78.56%	78.34%
Vessel F	Video 0	Fair	Moderate-easy	1415.7	72.07%	28.16%	31.44%	28.88%
	Video 1	Fair	Hard	695.3	84.48%	82.58%	82.20%	81.68%
	Video 2	Fair	Moderate	318.0	74.11%	80.59%	79.66%	77.82%
	Video 3	Good	Moderate	304.7	69.01%	70.20%	68.14%	65.37%
	Video 4	Good	Moderate-easy	225.0	79.98%	79.27%	78.54%	77.42%
	Video 5	Good	Hard	162.0	76.08%	73.06%	71.83%	70.59%
	Video 6	Good	Moderate	390.3	76.37%	73.73%	72.91%	71.62%
	Video 7	Good	Easy	80.0	88.41%	76.92%	76.36%	76.31%
	Video 8	Moderate	Hard	237.3	63.23%	63.47%	61.39%	58.95%
	Average			425.37037	75.97%	69.78%	69.17%	67.63%

Table 7.27: Per-video inter-observer summary for average of all Cefas observers (X, Y and Z), vessels E-F

Results overview

First we consider the count accuracy; the metric that is most useful to the marine biologists who are the intended users of our system.

The performance of the discard counter without species classification (in effect the combination of the detector, – or rather instance segmenter – tracker and discard counter) can be seen in the per-video summary in Table 7.17. Looking at the *average* row for each vessel we see that the count accuracy of these parts of the system ranges from 34.57% for vessel D to 77.16% for vessel F. We attribute the worse performance for vessel D to a combination of quality of the footage and the camera angle. Fine details were harder to see in the footage from vessel D as it was noticeably fuzzier than that of other vessels. Furthermore, the position of the camera seen in Figure 7.1 (d) and (e) resulted in fishers frequently obscuring the discard chute in Figure 7.1 (e) or the discard end of the belt in Figure 7.1 (d), where the chute is out of view. We observed fishers frequently activating the belt while leaning over it, obscuring the view of the discard region at the time when we need it to be visible for our system to successfully identify the fish that cross the discard threshold.

The performance of the discard counter when considering species classification can be seen in the *video count accuracy* column of the *average* rows of the per-species summary tables; Tables 7.10, 7.11, 7.12, 7.13, 7.14 and 7.15 for vessels A-F and Table 7.16 for all vessels together. The average count accuracy for any given species ranges from 13.21% for vessel B to 22.03% for vessel A, or 19.18% over all vessels. From the true and predicted counts in the *any species* rows of the aforementioned tables we observe that our model tends to over-count for vessels A-D and under-count for vessels E and F.

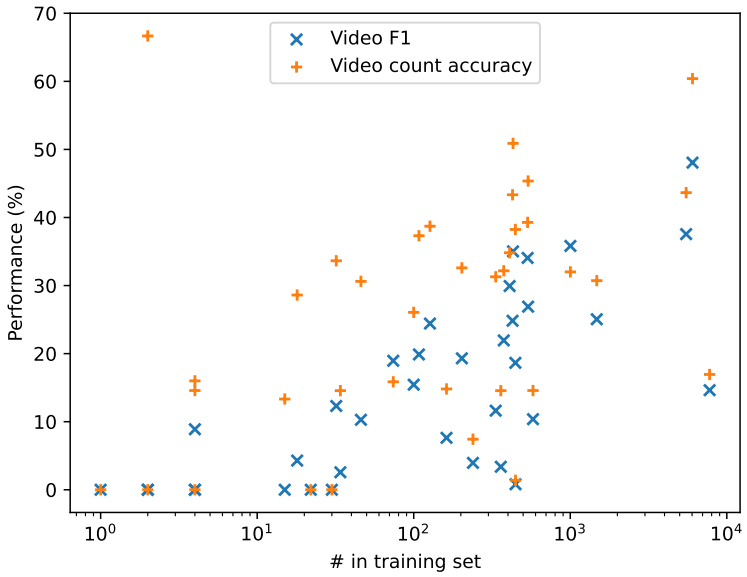


Figure 7.27: Relationship between per-species number of training samples and video count accuracy and F1-score. Note the log-scale on the horizontal axis.

The per-species *video count accuracy* and *video F1* columns in Table 7.16 (all vessels) shows – as expected – shows that accuracy generally improves with the number of samples in the training set. This is further illustrated in Figure 7.27. This highlights the necessity of sufficient representation of the classes that the system must identify.

The *video count accuracy* and *count accuracy* values are zero for species that are not represented in the training set or for species that are not present in the test set, but mis-predicted as being present. Otherwise they range from 1.39% for cod to 66.67% for catfish in the *video count accuracy* column and 1.72% for cod to 98.87% for bib in the *count accuracy* column.

We will now contrast the model performance confusion matrices in Figures 7.15, 7.16, 7.17, 7.18, 7.19 and 7.20 – for vessels A-F – with the inter-observer performance confusion matrices in Figures 7.21, 7.22, 7.23, 7.24, 7.25 and 7.26. For vessels E and F we observe much stronger diagonals – indicating more agreement – in the inter-observer performance confusion matrices than for those measuring the model performance. The comparatively weak diagonals in the confusion matrices measuring model performance for vessels E and F is partially explained by the tendency of the model to under-count. These false negatives manifest as higher figures in the right-most column in the extended confusion matrices. For vessels A-D the model achieves higher scores along the diagonal for some species than are present in the inter-observer confusion matrices. We attribute this to the tendency of our model to over-count; false positives can incorrectly match to ground truth discards inflating the scores along the diagonal. That said, while the scores can be lower, the diagonals in the inter-observer confusion matrices have fewer gaps. In summary, human observers exhibit stronger agreement when identifying the species of a discarded fish.

We now compare the per-species model performance summaries in Tables 7.10, 7.11, 7.12, 7.13, 7.14 and 7.15 with the inter-observer performance summaries in Tables 7.18, 7.19, 7.20, 7.21, 7.22 and 7.23. We observe that the inter-observer *video count accuracies* in the *any species* (that measures the accuracy of the overall count) rows are significantly better – usually approximately half the way closer to 100% – than the model count accuracies. Similarly the *video count accuracies* in the *average* row (the measures the average accuracy for any one species) are between 13.21% for to 22.03% for the model and between 29.98% and 45.47% for expert observers.

The aforementioned tendency of the model to strongly over-count in footage for vessels A-D and under-count in footage for vessels E and F is reflected in a comparison of the models' precision and recall scores against those of human observers. Focusing on the *any species* and *average* rows, the model achieves higher recall scores than human observers for vessels A-D and lower scores for vessels E and F. The counting errors result in the opposite trends being observed for the precision scores. It is for this reason that we chose to present F1 scores, in which we see human observers out-performing the model. The only exception to this are the F1 scores for the *any species* rows in vessels B and F.

Finally, we summarise our comparison of the performance of our model to that of human observers in Table 7.28.

Evaluate	Species	Video count acc	Video F1	Vessel count acc	Vessel F1
Model	Any species	60.90%	46.20%	70.31%	47.54%
	Average	19.18%	10.97%	36.57%	20.01%
MSS human observer	Any species	69.91%	44.05%	77.23%	45.20%
	Average	33.43%	22.40%	53.31%	28.85%
Cefas human observer	Any species	86.27%	73.96%	83.44%	68.22%
	Average	41.96%	39.26%	55.31%	45.44%

Table 7.28: Overall performance summary table comparing model performance to human observer performance

7.9 Discussion

We have discussed the development of a prototype system for analysing and quantifying by-catch from CCTV footage captured on fishing trawlers. Its components are able to operate on the challenging real-world footage obtained on board commercial fishing vessels. The major components of the system are in place.

The deep neural network models used in CatchMonitor require a significant amount of manually annotated training data in order to provide sufficient accuracy. The bottleneck imposed by the manual annotation process became apparent early in the project. This motivated us to explore semi-supervised learning and efficient annotation tools in order to alleviate it.

The core of the segmentation system is the well-known Mask R-CNN instance segmentation algorithm [He et al., 2017]. It's effectiveness relies on having a sufficiently large training set. Manually annotating the $\tilde{3}$,400 in our training set – see Table 7.2 – involved a considerable amount of effort on the part of our annotators at Marine Scotland Science and Cefas. While we reported positive semi-supervised learning results for the related problem of semantic segmentation in Chapter 5, our attempts to adapt our approach for instance segmentation did not yield a noticeable performance improvement. As an alternative method of reducing annotation effort, we adopted three approaches for improving our annotation tools (see Section 7.3.3). Firstly we implemented Boolean operations to support the user in quickly adding or removing parts from a label. Second we used a model trained on images annotated so far to predict instance labels for un-annotated images, allowing the user to fix mistakes or add missing labels rather than starting from scratch. Finally we implemented a tool that uses the DEXTR algorithm [Maninis et al., 2018] to automatically outline a fish given four points that identify its location and extents.

We chose to classify the species of individual fish using a separate classifier, rather than adopting the standard practice of building this functionality into the object detection part of the instance segmentation system. This allows us to horizontally align the randomly oriented fish that were detected in our footage. We found that a standard residual network based image classifier was adequate for species classification. We were also able to successfully

apply semi-supervised learning to this problem, adopting an approach very similar to our domain adaptation approach used in Chapter 4. We drove consistency regularization using CowMask [French. et al., 2022] and RandAugment [Cubuk et al., 2020; Xie et al., 2019; Sohn et al., 2020], finding that RandAugment delivers superior performance. We also found that our approach is able to reduce the effect of the domain gap (see Table 7.7), allowing footage captured on a research vessel to be used to train a classifier that can be applied to footage captured on-board commercial vessels. Annotation effort can be substantially reduced by using semi-artificial training data obtained by capturing images of fish of a known species on-board a research vessel or a helpful and co-operative commercial vessel. We consider our successful application of semi-supervised learning and domain adaptation to a challenging real-world image classification problem to be an interesting and valuable data point, as it strongly suggests that semi-supervised learning is ready for real-world use.

While promising, we consider the weak performance of our species classifier on a subset of the species present to be an outstanding issue in our system. We attribute this to poor representation of those species in our dataset. Expanding a dataset such as ours to improve the representation of these species is challenging as additional commercial footage would likely have a similar species distribution to our existing data. Footage obtained from a research vessel could partially alleviate this by presenting individuals of under-represented species multiple times in a variety of angles. That said, research vessel footage can only feature individuals of species that the vessel staff were able to catch. Images of fish from rare or difficult to find species will be hard to obtain, so increasing representation of those species could be expensive.

Our approach for object tracking and discard counting is simple and effective. Bewley *et al.* noted that the performance the SORT [Bewley et al., 2016] tracker is heavily dependent on and therefore limited by the effectiveness of their object detector. Given that we adopt a very similar approach our tracker suffers from similar limitations.

Following Bewley et al. [2016] we used a Kalman filter to model the motion of tracked fish. The simple linear dynamics modeled by a Kalman filter do not precisely model the motion present in our problem domain. The discrete application of the filter does not account for varying video frame rates; a continuous time Kalman filter [Brown and Hwang, 1997; Bar-Shalom et al., 2004] would account for these variations. Extended Kalman filters [Sorenson, 1985] and unscented Kalman Filters [Julier and Uhlmann, 2004] model non-linear dynamics and non-linear correspondence between the state and observation. These variants of Kalman filtering would allow more accurate modelling of the motion observed in our CCTV footage, in addition to accounting for the non-linear camera projection.

We also observed that in some of our CCTV footage the fishers would obscure the portion of the conveyor belt in immediate proximity to the discard chute. In such situations the fish are not detected and the discards are not counted. While we have attempted to develop our system so as to avoid imposing changes to the working practices on-board commercial fishing vessels as much as possible, we feel that some changes may be required to attain sufficient accuracy. The effectiveness of our system would be maximized by placing the camera directly above a 30cm - 50cm region of the belt in proximity to the discard chute and requiring that fishers should avoid working on or obscuring the view of this region of the

belt. Furthermore, manual analysis of the video output generated by our discard counter indicated that higher video frame rates improve the accuracy of our tracker. We would therefore suggest a frame rate of 25 frames per second or above; the videos that we used had frame rates of either 5 or 15 frames per second.

The results presented in Section 7.8.6 are not sufficiently accurate for use by Marine biologists who would require count accuracies for most species to be 90% or greater. None-theless our prototype system shows promise as accuracy would likely improve with additional training data, videos with higher frame rates and the changes to working practices discussed above. It is especially worth noting that the test data against which we evaluated our system featured species that are not found in the training set, clearly indicating that further training data is required.

In general we observe that expert observers exhibit strong agreement with one another as to the species of discarded fish. Our species classifier does not match the performance of a typical expert observer.

The main disagreement between expert observers arises from the choice of which fish to count as discards and which to ignore. The criteria employed by an observer to decide as to whether a discard should be counted or not can include its size; smaller fish may be ignored by some observers. Furthermore in our footage we observed fishers cutting open the stomachs of large fish, resulting in a number of smaller fish that it had eaten being disgorged onto the belt. Expert observers would not normally count these smaller fish as discards. Our system does not detect such situations. While it could be extended to recognize these scenarios and quantify them accordingly, we believe that this would be a non-trivial task.

Computer vision is a fast moving field; future developments and improved techniques can be incorporated into the relevant components of our system in the future with a view to improving performance. For example, the recently proposed ByteTrack [Zhang et al., 2021] is a simple modification to the SORT object tracking algorithm that offers improved tracking accuracy.

The recently proposed Segment Anything model [Kirillov et al., 2023] is a vision transformer [Dosovitskiy et al., 2021] based foundation model trained on over 100 million images for the task of instance segmentation. Preliminary experiments indicated that its speed was significantly below that of Mask R-CNN, leading us to consider it to be a worthy replacement for the Mask R-CNN and DEXTR based automated segmentation approach used to generate automatically created initial labels to be edited within our labelling tool. Future speed improvements could result in such a model being a viable candidate for replacing Mask R-CNN completely.

OmniMotion [Wang et al., 2023], TAPIR [Doersch et al., 2023] and CoTracker [Karaev et al., 2023] are recent point tracking approaches that demonstrate occlusion resistance and strong performance. The run-time of the optimization process used by OmniMotion would limit is application to that of a pre-process used for generating training data for other more performant approaches. While these approaches appear promising, initial experiments with TAPIR and CoTracker indicated that they are not well suited to the low frame rate CCTV

footage present in our dataset. They were able to follow the overall motion of the conveyor belt but did not track points placed on individual fish that were manipulated by fishers. While not yet applicable to our problem domain, they represent a fascinating line of research that should be followed.

We also note that [Strachan, 1993] found that seasonal changes in the physical condition of fish and variability in their colour hampered the effectiveness of their approach. Such sources of variation are likely to affect our system as well. The effects of seasonal and individual variation need to be assessed in order to ensure reliable performance in the field.

In summary, we have presented a prototype solution to automated by-catch quantification. Our approach combines a well known and reliable components – namely Mask R-CNN instance segmentation, image classification and SORT object tracking – with novel semi-supervised learning approaches for improving the accuracy of species identification. We can conclude that the use of computer vision to quantify fish discards from surveillance footage is feasible with current state-of-the-art algorithms.

8 Conclusions

8.1 Contributions

In this thesis we have made five main contributions that we will now discuss.

8.1.1 Multi-spectral object detection

In Chapter 3 we contributed an exploration of the use of spectral edge image fusion to fuse the thermal band of a multi-spectral image into the RGB band as a pre-process, prior to use as an input to an object detection neural network model. While fusing thermal information into an RGB image improves accuracy, it is not as effective as providing the thermal band as a fourth input channel to the network and allowing the network to learn to utilize it.

As shown in Chapter 3 and in the work of Lui Jingjing Liu and Metaxas [2016], Hwang Hwang et al. [2015] and Konig König et al. [2017], multi-spectral imagery can improve the accuracy of computer vision models. The use of either multi-spectral imagery or additional data sources – such as depth imagery – could provide a valuable signal for both segmentation and species classification.

The issues likely to complicate the use of multi-spectral imagery for fisheries surveillance are practical in nature. As shown in Section 3.5, imperfect alignment of the different image channels can affect the performance of the model. The installation of a combined RGB and thermal camera rig similar to that used for the capture of the KAIST [Hwang et al., 2015] dataset over the conveyor belt within a fishing trawler would likely be difficult. Fishing trawlers often provide cramped working conditions, so space is at a premium, limiting the size of camera installations. Furthermore, the rolling of the ocean and any physical jolts to the rig could knock the optical components of the rig out of alignment, likely exacerbating the problems observed in the KAIST dataset. This would suggest that utilizing multi-spectral imagery would be best accomplished using a multi-spectral camera whose pixels feature more than 3 channels. Such devices are uncommon and are likely to be expensive.

Utilizing depth cameras was considered early during the SMARTFISH project. This possibility was rejected due to limitations of the technology available at the time (early 2018); choosing a camera and depth capture technology necessitated a trade-off between frame-rate, depth resolution, depth map resolution and material limitations (reflective surfaces such as fish scales could cause some depth capture approaches to fail). During the last three years, this situation has improved as such trade-offs are no longer necessary. This is an option that would be worth exploring in future work.

8.1.2 Consistency regularization for visual domain adaptation

In Chapter 4 we contributed a consistency regularization based domain adaptation algorithm that achieved state of the art results in small image benchmarks and the VisDA-17 domain adaptation challenge [Peng et al., 2018]. We observed the similarity between semi-supervised learning and domain adaptation problems, and adapted the strong performing Mean Teacher [Tarvainen and Valpola, 2017] semi-supervised classification algorithm. Our results across multiple datasets and network architectures further confirm the effectiveness and wide applicability of consistency regularization. Our work in this chapter formed the foundation of much of our semi-supervised learning experiments conducted afterwards, that we discuss in Chapters 5 and 7.

Our success in this work led us to anticipate that relatively simple adaptations would allow us to use it in CatchMonitor for semantic segmentation and later for species identification. Our early experiments however were unsuccessful. Our lack of success in semantic segmentation motivated us to explore this problem in more detail, which we covered in Chapter 5. While our early semi-supervised species classification experiments were unsuccessful, we later found that stronger augmentation was required for this challenging real-world problem.

Our strong results on the challenging MNIST→SVHN domain adaptation pathway were enabled by the use of a customized augmentation scheme. This augmentation scheme was designed specifically for this problem. While it is therefore not widely applicable, it demonstrated the importance of strong augmentation for the successful application of consistency regularization based semi-supervised learning algorithms. We note that strong augmentation schemes such as RandAugment [Cubuk et al., 2020] have proved to be highly effective for semi-supervised learning [Xie et al., 2019; Sohn et al., 2020]. A worthwhile avenue would be to replace the augmentation schemes used in Chapter 4 with RandAugment and contrast its effectiveness the existing results.

We note that our use of confidence thresholding has since been adopted by more recent work Sohn et al. [2020].

8.1.3 CutMix and colour augmentation for semi-supervised semantic segmentation

In Chapter 5 we analysed the problem of semi-supervised semantic segmentation. We proposed two aspects of the semantic segmentation problem that complicate semi-supervised learning when using consistency regularization. Firstly we analysed the input data distribution and found that the cluster assumption does not apply as there are no low-density regions separating clusters of samples belonging to different classes. A number of prior publications in the area of semi-supervised classification stated that the cluster assumption is key to success. Secondly we found that the network could minimize the consistency loss term that we used in our approach could be minimized by learning to predict the class based solely on the colour of the corresponding input pixel, ignoring the surrounding context. We were able to verify this by applying colour augmentation (inspired by the ablation conducted in the self-supervised work of Chen et al. [2020a]), finding that this permitted

standard affine augmentation driven consistency regularization to result in successful semi-supervised learning on segmentation datasets such as PASCAL and ISIC 2017. We adapted the CutMix regularizer of Yun et al. [2019], yielding state of the art semi-supervised semantic segmentation results across a number of benchmarks. CutMix proved to be sufficiently effective to operate without requiring colour augmentation.

Given that we found that the cluster assumption does not apply in semantic segmentation problems, our results and other results in the literature suggest that the cluster assumption is in fact *not* a pre-requisite for successful semi-supervised learning. This bolsters our assessment of consistency regularization, suggesting that it is a very powerful and flexible regularizer that operates in challenging conditions across a variety of problem domains. Furthermore, the challenging nature of semantic segmentation suggests that it can be used as an *acid test* for evaluating future semi-supervised or unsupervised regularizers, as the regularizer cannot utilize low-density regions in the input distribution to guide it.

8.1.4 Milking CowMask for semi-supervised image classification.

In Chapter 6 we present a mask-based regularizer that we call CowMask. It combines mask-based regularization [DeVries and Taylor, 2017], mix-based regularization [Verma et al., 2019] and Mean Teacher [Tarvainen and Valpola, 2017] with the structure of FixMatch [Sohn et al., 2020]. We achieved a state of the art result for 10% semi-supervised ImageNet in mid-January 2020 using an approach that is quite simple in comparison to other contemporary approaches. In mid-February 2020 our results were beaten by those of SimCLR [Chen et al., 2020a], demonstrating the highly competitive nature of the field.

In summary, we developed CowMask and milked it for semi-supervised image classification[1]. We hope that it is a useful contribution to the *field*. In the context of CatchMonitor however, our semi-supervised species classification results in Chapter 7 show RandAugment [Cubuk et al., 2020] providing superior performance to CowMix. We therefore hope that we do not have to put CowMask out to pasture.

8.1.5 CatchMonitor: an automated CCTV analysis system

Building on our experience gained during the work covered in the previous chapters, we developed an automated by-catch quantification system called CatchMonitor, described in Chapter 7. The majority of CatchMonitor uses and builds on well established techniques that have proved to be effective in practice; we use Mask R-CNN [He et al., 2017] for instance segmentation, our ResNet-50 [He et al., 2016] based supervised species classifier is trained using transfer learning and our annotation tool simplifies instance annotation using an implementation of DEXTR [Maninis et al., 2018] and by using predictions from a previously trained model as a starting point. In spite of the effective techniques listed above, the manual annotation process required to create the training data proved to be a bottleneck; this motivated us to explore the semi-supervised and domain adaptation work discussed in Chapters 4 and 5.

[1] We also milked it for cheesy puns

While our attempts at adapting our semi-supervised semantic segmentation algorithm for instance segmentation were not successful, our semi-supervised classification results were very positive. The approach used for species identification shares many similarities with our domain adaptation algorithm from Chapter 4, with some advancements drawn from other work in the field (namely Sohn et al. [2020], Xie et al. [2019] and Cubuk et al. [2020]). In effect we adopt the CowOut semi-supervised classification approach developed in Chapter 6 and replace CowOut with RandAugment. Automated by-catch quantification from CCTV footage is a challenging real-world problem that involves processing imagery captured under decidedly non-ideal conditions. Our success in applying semi-supervised classification algorithms in this domain gives a strong suggestion that these approaches are ready – or nearly ready – for widespread use on practical and commercial computer vision problems.

Our discard counter adapts the simple and effective SORT [Bewley et al., 2016] algorithm to track individual fish throughout a video and determine those that are discarded. We were able to develop a prototype system that achieves the goal of the project; namely video in, by-catch quantification out. We applied our working prototype to a number of videos that have been analysed by expert observers. While our results suggest that CatchMonitor is not yet ready for real-world use, in order to give a realistic assessment of the real-world performance of our system we adopted a very exacting evaluation methodology, giving low performance scores for species that were not included in the training set. Expanding the training set to provide sufficient coverage for all species that we wish to quantify in addition to featuring footage from a more diverse range of vessels would likely improve performance significantly. Furthermore, future developments in the computer vision field may arise in improved approaches that could further improve the performance of the components of CatchMonitor. We believe that such developments could improve CatchMonitor to the point of being able to provide automated by-catch quantification of sufficient quality for the fishing sector.

8.2 Thoughts on semi-supervised learning

Aitchison [2020] notes that much prior work in the field of semi-supervised learning [Sohn et al., 2020; Xie et al., 2019] focuses on standard datasets whose ground truths were the result of agreement between multiple human annotators. Their construction explicitly excludes samples that are either ambiguous – whose ground truth cannot be reliably determined – or unidentifiable from the dataset. This improves the accuracy of the ground truth labels, resulting in more reliable metrics and improved repeatability. However, it results in a dataset that is less representative of real-world practical scenarios. Annotating a dataset using multiple human annotators increases cost, excluding smaller organisations with a limited annotation budget. This also precludes evaluating a model on ambiguous or unidentifiable samples that will be typical of practical problems.

The issues described above precisely describe the challenges we faced in the CatchMonitor project. The cost of annotation precluded us from using multiple human annotators. Furthermore, many fish isolated by the segmenter are very challenging or impossible to classify

by species due to occlusions, orientation or poor visibility. The segmenter can also misidentify objects on the conveyor belt as fish, resulting in out-of-distribution samples.

We hypothesize that the rigorous dataset annotation and construction approaches used in the creation of standard datasets could have the effect of imposing the cluster assumption upon the data distribution. Eliminating ambiguous or out of distribution samples would decrease the density of samples between classes, resulting in low-density regions. With a view to maximizing practical applicability we suggest evaluating the performance of semi-supervised learning algorithms on datasets more typical of real-world problems – such as Van Horn et al. [2018] – in the future. We would also suggest carefully evaluating existing datasets and developing new datasets that explicitly feature ambiguous and out-of-distribution samples in order to assess the performance classification algorithms – both supervised and semi-supervised – in situations that are representative of real-world commercial conditions.

9 Bibliography

Aitchison, L. (2020). A statistical theory of semi-supervised learning. *arXiv preprint arXiv:2008.05913*.

Alsmadi, M. K. S., Omar, K. B., Noah, S. A., and Almarashdah, I. (2009). Fish recognition based on the combination between robust feature selection, image segmentation and geometrical parameter techniques using artificial neural network and decision tree. *International Journal of Computer Science and Information Security*, 2(2):215–221.

Athiwaratkun, B., Finzi, M., Izmailov, P., and Wilson, A. G. (2019). There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations*.

Azizi, S., Mustafa, B., Ryan, F., Beaver, Z., Freyberg, J., Deaton, J., Loh, A., Karthikesalingam, A., Kornblith, S., Chen, T., Natarajan, V., and Norouzi, M. (2021). Big self-supervised models advance medical image classification.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561.

Bai, M. and Urtasun, R. (2017). Deep watershed transform for instance segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2858–2866. IEEE.

Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2004). *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

Bergmann, P., Meinhardt, T., and Leal-Taixe, L. (2019). Tracking without bells and whistles. In *Proceedings of the IEEE international conference on computer vision*, pages 941–951.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. (2019a). Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. (2019b). Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5050–5060.

Beucher, S. and Meyer, F. (1993). The morphological approach to segmentation: the watershed transformation. Mathematical morphology in image processing. *Optical Engineering*, 34:433–481.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE.

Beyer, L., Zhai, X., and Kolesnikov, A. (2022). Better plain ViT baselines for imagenet-1k. *arXiv preprint arXiv:2205.01580*.

Boom, B. J., Huang, P. X., He, J., and Fisher, R. B. (2012). Supporting ground-truth annotation of image datasets using clustering. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1542–1545. IEEE.

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. (2017). Unsupervised pixel-level domain adaptation with generative adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 95–104.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016). Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351.

Bowler, E., Fretwell, P. T., French, G., and Mackiewicz, M. (2019). Using deep learning to count albatrosses from space. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 10099–10102. IEEE.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Brown, R. G. and Hwang, P. Y. (1997). *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions*. Wiley.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Chapelle, O. and Zien, A. (2005). Semi-supervised classification by low density separation. In *AISTATS*, volume 2005, pages 57–64. Citeseer.

Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(3).

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607. PMLR.

Chen, T., Luo, C., and Li, L. (2021). Intriguing properties of contrastive losses. *Advances in Neural Information Processing Systems*, 34:11834–11845.

Chen, X., Fan, H., Girshick, R., and He, K. (2020b). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Chen, X., Hsieh, C.-J., and Gong, B. (2022). When vision transformers outperform resnets without pre-training or strong data augmentations. In *International Conference on Learning Representations*.

Chintala, S. et al. (2017). Pytorch.

Choi, W. and Savarese, S. (2010). Multiple target tracking in world coordinate with single, minimally calibrated camera. In *European Conference on Computer Vision*, pages 553–567. Springer.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

Chuang, M.-C., Hwang, J.-N., and Williams, K. (2016). A feature learning and object recognition framework for underwater fish images. *IEEE Transactions on Image Processing*, 25(4):1862–1872.

Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., et al. (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172. IEEE.

Connah, D., Drew, M. S., and Finlayson, G. D. (2015). Spectral edge: gradient-preserving spectral mapping for image fusion. *JOSA A*, 32(12):2384–2396.

Connelly, S. (2017). polybooljs.

Csurka, G., Baradel, F., Chidlovskii, B., and Clinchant, S. (2017). Discrepancy-based networks for unsupervised domain adaptation: A comparative study. In *The IEEE International Conference on Computer Vision (ICCV)*.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624.

Cui, W., Liu, Y., Li, Y., Guo, M., Li, Y., Li, X., Wang, T., Zeng, X., and Ye, C. (2019). Semi-supervised brain lesion segmentation with an adapted mean teacher model. In *International Conference on Information Processing in Medical Imaging*, pages 554–565. Springer.

Dai, J., He, K., Li, Y., Ren, S., and Sun, J. (2016a). Instance-sensitive fully convolutional networks. In *European Conference on Computer Vision*, pages 534–549. Springer.

Dai, J., He, K., and Sun, J. (2016b). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017a). Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773.

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. R. (2017b). Good semi-supervised learning that requires a bad gan. In *Advances in neural information processing systems*, pages 6510–6520.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552.

Doersch, C., Yang, Y., Vecerik, M., Gokay, D., Gupta, A., Aytar, Y., Carreira, J., and Zisserman, A. (2023). Tapir: Tracking any point with per-frame initialization and temporal refinement. *arXiv preprint arXiv:2306.08637*.

Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1532–1545.

Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2009). Pedestrian detection: A benchmark. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2011). Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761.

Dollár, P. and Zitnick, C. L. (2013). Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Finlayson, G. D., Connah, D., and Drew, M. S. (2011). Lookup-table-based gradient field reconstruction. *IEEE transactions on image processing*, 20(10):2827–2836.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2021). Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*.

French, G., Finlayson, G., and Mackiewicz, M. (2018a). Multi-spectral pedestrian detection via image fusion and deep neural networks. *Color and Imaging Conference*, 2018(1):176–181.

French, G., Fisher, M., Mackiewicz, M., and Needle, C. (2015). Convolutional neural networks for counting fish in fisheries surveillance video.

French, G., Laine, S., Aila, T., Mackiewicz, M., and Finlayson, G. (2020a). Semi-supervised semantic segmentation needs strong, varied perturbations. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press.

French., G. and Mackiewicz., M. (2022). Colour augmentation for improved semi-supervised semantic segmentation. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP,*, pages 356–363. INSTICC, SciTePress.

French, G., Mackiewicz, M., and Fisher, M. (2018b). Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*.

French, G., Mackiewicz, M., Fisher, M., Holah, H., Kilburn, R., Campbell, N., and Needle, C. (2020b). Deep neural networks for analysis of fisheries surveillance video and automated monitoring of fish discards. *ICES Journal of Marine Science*, 77(4):1340–1353.

French., G., Oliver., A., and Salimans., T. (2022). Milking cowmask for semi-supervised image classification. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*, pages 75–84. INSTICC, SciTePress.

Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., and Berg, A. C. (2017). Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*.

Ganin, Y. and Lempitsky, V. (2014). N^4-fields: Neural network nearest neighbor fields for image transforms. In *Asian Conference on Computer Vision*, pages 536–551. Springer.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1180–1189.

Gastaldi, X. (2017). Shake-shake regularization. *arXiv preprint arXiv:1705.07485*.

Ghifary, M., Kleijn, W. B., Zhang, M., Balduzzi, D., and Li, W. (2016). Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer.

Girshick, R. (2015). Fast r-cnn. In *International Conference on Computer Vision*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Guerrero-Pena, F. A., Fernandez, P. D. M., Ren, T. I., Yui, M., Rothenberg, E., and Cunha, A. (2018). Multiclass weighted loss for instance segmentation of cluttered cells. *arXiv preprint arXiv:1802.07465*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777.

Haeusser, P., Frerix, T., Mordvintsev, A., and Cremers, D. (2017). Associative domain adaptation. In *International Conference on Computer Vision (ICCV)*.

Han, S., Huang, P., Wang, H., Yu, E., Liu, D., Pan, X., and Zhao, J. (2020). Mat: Motion-aware multi-object tracking. *arXiv preprint arXiv:2009.04794*.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. (2011). Semantic contours from inverse detectors. In *International Conference on Computer Vision*, pages 991–998.

Harris, C. G., Stephens, M., et al. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer.

Harris, E., Marcu, A., Painter, M., Niranjan, M., Prügel-Bennett, A., and Hare, J. (2020). Fmix: Enhancing mixed sample data augmentation. *arXiv preprint arXiv:2002.12047*.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

He, K., Girshick, R., and Dollár, P. (2018). Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European conference on computer vision*, pages 346–361. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hoffer, E. and Ailon, N. (2016). Semi-supervised deep learning by metric embedding. *arXiv preprint arXiv:1611.01449*.

Hu, B. G., Gosine, R., Cao, L. X., and de Silva, C. (1998). Application of a fuzzy classification technique in computer grading of fish products. *Fuzzy Systems, IEEE Transactions on*, 6(1):144–152.

Hu, J., Li, D., Duan, Q., Han, Y., Chen, G., and Si, X. (2012). Fish species classification by color, texture and multi-class support vector machine using computer vision. *Comput. Electron. Agric.*, 88:133–140.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. (2017). Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1558–1567.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Hung, W.-C., Tsai, Y.-H., Liou, Y.-T., Lin, Y.-Y., and Yang, M.-H. (2018). Adversarial learning for semi-supervised semantic segmentation. *CoRR*, abs/1802.07934.

Hwang, S., Park, J., Kim, N., Choi, Y., and So Kweon, I. (2015). Multispectral pedestrian detection: Benchmark dataset and baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1037–1045.

Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ji, X., Henriques, J. F., and Vedaldi, A. (2019). Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874.

Jingjing Liu, Shaoting Zhang, S. W. and Metaxas, D. (2016). Multispectral deep neural networks for pedestrian detection. In Richard C. Wilson, E. R. H. and Smith, W. A. P., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 73.1–73.13. BMVA Press.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kalluri, T., Varma, G., Chandraker, M., and Jawahar, C. (2018). Universal semi-supervised semantic segmentation. *CoRR*, abs/1811.10323.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., and Rupprecht, C. (2023). Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635*.

Karthik, S., Prabhu, A., and Gandhi, V. (2020). Simple unsupervised multi-object tracking. *arXiv preprint arXiv:2006.02609*.

Ke, R., Aviles-Rivero, A. I., Pandey, S., Reddy, S., and Schönlieb, C.-B. (2022). A three-stage self-training framework for semi-supervised semantic segmentation. *IEEE Transactions on Image Processing*, 31:1805–1815.

Ke, Z., Wang, D., Yan, Q., Ren, J., and Lau, R. W. (2019). Dual student: Breaking the limits of the teacher in semi-supervised learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6728–6736.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *stat*, 1050:10.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. (2023). Segment anything. *arXiv preprint arXiv:2304.02643*.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016). Jupyter notebooks-a publishing format for reproducible computational workflows. In *20th International Conference on Electronic Publishing*, pages 87–90.

König, D., Adam, M., Jarvers, C., Layher, G., Neumann, H., and Teutsch, M. (2017). Fully convolutional region proposal networks for multispectral person detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 49–56.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*.

Laine, S. and Aila, T. (2017). Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*.

Li, J., Xiong, C., and Hoi, S. C. (2021). Comatch: Semi-supervised learning with contrastive graph regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9475–9484.

Li, X., Chen, H., Qi, X., Dou, Q., Fu, C.-W., and Heng, P.-A. (2018a). H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *IEEE transactions on medical imaging*, 37(12):2663–2674.

Li, X., Yu, L., Chen, H., Fu, C.-W., and Heng, P.-A. (2018b). Semi-supervised skin lesion segmentation via transformation consistent self-ensembling model. In *British Machine Vision Conference*.

Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. (2016). Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. (2017). Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2018). Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European conference on computer vision*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Liu, Y., Yao, J., Li, L., Lu, X., and Han, J. (2017). Learning to refine object contours with a top-down fully convolutional encoder-decoder network. *arXiv preprint arXiv:1705.04456*.

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., and Guo, B. (2022). Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12009–12019.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Long, J., Shelhamer, E., and Darrell, T. (2015a). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

Long, M., Cao, Y., Wang, J., and Jordan, M. (2015b). Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lundh, F., Clark, A., et al. (2020). Pillow.

Luo, Y., Zhu, J., Li, M., Ren, Y., and Zhang, B. (2018). Smooth neighbors on teacher graphs for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8896–8905.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1445–1453.

Maninis, K.-K., Caelles, S., Pont-Tuset, J., and Van Gool, L. (2018). Deep extreme cut: From extreme points to object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 616–625.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423.

Martin Arjovsky, S. and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia*.

Mathiassen, J. R., Misimi, E., Bondø, M., Veliyulin, E., and Østvik, S. O. (2011). Trends in application of imaging technologies to inspection of fish and fish products. *Trends in Food Science & Technology*, 22(6):257 – 275.

Mittal, S., Tatarchenko, M., and Brox, T. (2019a). Semi-supervised semantic segmentation with high-and low-level consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Mittal, S., Tatarchenko, M., Çiçek, Ö., and Brox, T. (2019b). Parting with illusions about deep active learning. *arXiv preprint arXiv:1912.05361*.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2017). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*.

Needle, C. L., Dinsdale, R., Buch, T. B., Catarino, R. M. D., Drewery, J., and Butler, N. (2014). Scottish science applications of remote electronic monitoring. *ICES Journal of Marine Science: Journal du Conseil*.

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., and Goodfellow, I. J. (2018). Realistic evaluation of semi-supervised learning algorithms. In *International Conference on Learning Representations*.

Olsson, V., Tranheden, W., Pinto, J., and Svensson, L. (2021). Classmix: Segmentation-based data augmentation for semi-supervised learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1369–1378.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Ouali, Y., Hudelot, C., and Tami, M. (2020). An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*.

Peng, X., Usman, B., Kaushik, N., Wang, D., Hoffman, J., and Saenko, K. (2018). Visda: A synthetic-to-real benchmark for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2021–2026.

Perez, F., Vasconcelos, C., Avila, S., and Valle, E. (2018). Data augmentation for skin lesion analysis. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 303–311. Springer.

Perone, C. S. and Cohen-Adad, J. (2018). Deep semi-supervised segmentation with weight-averaged consistency targets. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 12–19. Springer.

Pham, H., Dai, Z., Xie, Q., and Le, Q. V. (2021). Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568.

Pinheiro, P. O., Collobert, R., and Dollár, P. (2015). Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998.

Pinheiro, P. O., Lin, T.-Y., Collobert, R., and Dollár, P. (2016). Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Qin, H., Li, X., Liang, J., Peng, Y., and Zhang, C. (2016). Deepfish: Accurate underwater live fish recognition with a deep architecture. *Neurocomputing*, 187:49–58.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.

Real, E., Shlens, J., Mazzocchi, S., Pan, X., and Vanhoucke, V. (2017). Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 7464–7473. IEEE.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28*, pages 91–99.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

Rosenfeld, A., Zemel, R., and Tsotsos, J. K. (2018). The elephant in the room. *arXiv preprint arXiv:1808.03305*.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Russell, C., Kohli, P., Torr, P. H., et al. (2009). Associative hierarchical CRFs for object class image segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 739–746. IEEE.

Russo, P., Carlucci, F. M., Tommasi, T., and Caputo, B. (2018). From source to target and back: symmetric bi-directional adaptive gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8099–8108.

Saito, K., Ushiku, Y., and Harada, T. (2017a). Asymmetric tri-training for unsupervised domain adaptation. *arXiv preprint arXiv:1702.08400*.

Saito, K., Ushiku, Y., Harada, T., and Saenko, K. (2017b). Adversarial dropout regularization. *CoRR*, abs/1711.01575.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016a). Mutual exclusivity loss for semi-supervised deep learning. In *23rd IEEE International Conference on Image Processing, ICIP 2016*. IEEE Computer Society.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016b). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234.

Sankaranarayanan, S., Balaji, Y., Castillo, C. D., and Chellappa, R. (2017). Generate to adapt: Aligning domains using generative adversarial networks. *arXiv preprint arXiv:1704.01705*.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Shen, W., Wang, X., Wang, Y., Bai, X., and Zhang, Z. (2015). Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3982–3991.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2242–2251.

Shu, R., Bui, H., Narui, H., and Ermon, S. (2018). A DIRT-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, volume 33, pages 596–608.

Sorenson, H. W. (1985). *Kalman filtering: theory and application*. IEEE.

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. *stat*, 1050:19.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460.

Stekovic, S., Fraundorfer, F., and Lepetit, V. (2018). S4-net: Geometry-consistent semi-supervised semantic segmentation. *CoRR*, abs/1812.10717.

Storbeck, F. and Daan, B. (2001). Fish species recognition using computer vision and a neural network. *Fisheries Research*, 51(1):11 – 15.

Strachan, N. J. C. (1993). Recognition of fish species by colour and shape. *Image and Vision Computing*, pages 2–10.

Sun, B. and Saenko, K. (2016). Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops*, pages 443–450.

Sun, X., Shi, J., Dong, J., and Wang, X. (2016). Fish recognition from low-resolution underwater images. In *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 471–476. IEEE.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204.

Tatem, A. J., Goetz, S. J., and Hay, S. I. (2008). Fifty years of earth observation satellites. *American scientist*, 96(5):390–398.

Tayama, I., Shimdate, M., Kubuta, N., and Nomura, Y. (1982). Application of optical sensor for fish sorting. *Refigeration*, 57(661):1146–1150.

The Nature Conservancy (2016). The Nature Conservancy Fisheries Monitoring Competition at Kaggle.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. *arXiv preprint arXiv:1702.05464*.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. (2014). Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Uhrig, J., Cordts, M., Franke, U., and Brox, T. (2016). Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., and Torr, P. H. (2017). End-to-end representation learning for correlation filter based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2805–2813.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors (2014). scikit-image: image processing in Python. *PeerJ*, 2:e453.

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. (2018). The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. (2019). Interpolation consistency training for semi-supervised learning. *CoRR*, abs/1903.03825.

Wang, D. and Shang, Y. (2014). A new active labeling method for deep learning. In *IJCNN*.

Wang, F., Kong, T., Zhang, R., Liu, H., and Li, H. (2021). Self-supervised learning by estimating twin class distributions. *arXiv preprint arXiv:2110.07402*.

Wang, G., Wang, Y., Zhang, H., Gu, R., and Hwang, J.-N. (2019a). Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 482–490.

Wang, J., He, Z., Feng, C., Zhu, Z., Lin, Q., Lv, J., and Xie, S. (2018). Domain confusion with self ensembling for unsupervised adaptation. *arXiv preprint arXiv:1810.04472*.

Wang, Q., Chang, Y.-Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., and Snavely, N. (2023). Tracking everything everywhere all at once.

Wang, X., Kihara, D., Luo, J., and Qi, G.-J. (2019b). EnAET: Self-trained ensemble autoencoding transformations for semi-supervised learning. *arXiv preprint arXiv:1911.09265*.

White, D. J., White, C. J., Svellingen, C., and Strachan, N. C. J. (2006). Automated measurement of species and length of fish by computer vision. *Fisheries Research*, 80:203–210.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*.

Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403.

Yuan, J., Liu, Y., Shen, C., Wang, Z., and Li, H. (2021). A simple baseline for semi-supervised semantic segmentation with strong data augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8229–8238.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In Richard C. Wilson, E. R. H. and Smith, W. A. P., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press.

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. (2019). S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1476–1485.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Zhang, L., Lin, L., Liang, X., and He, K. (2016). Is faster r-cnn doing well for pedestrian detection? In *European conference on computer vision*, pages 443–457. Springer.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2021). Bytetrack: Multi-object tracking by associating every detection box. *arXiv preprint arXiv:2110.06864*.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537.

Zheng, Z., Guo, C., Zheng, X., Yu, Z., Wang, W., Zheng, H., Fu, M., and Zheng, B. (2018). Fish recognition from a vessel camera using deep convolutional neural network and data augmentation. In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–5. IEEE.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation.

Zhou, X., Koltun, V., and Krähenbühl, P. (2020). Tracking objects as points. *arXiv preprint arXiv:2004.01177*.

Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*.

Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2021). Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*.

Zion, B. (2012). The use of computer vision technologies in aquaculture - A review. *Computers and Electronics in Agriculture*, 88:125–132.

A Appendix: Semi-supervised semantic segmentation 2D toy experimnent setup

The neural networks used in our 2D toy experiments are simple classifiers in which samples are 2D x, y points ranging from -1 to 1. Our networks are multi-layer perceptrons consisting of 3 hidden layers of 512 units, each followed by a ReLU non-linearity. The final layer is a 2-unit classification layer. We use the mean teacher Tarvainen and Valpola [2017] semi-supervised learning algorithm with binary cross-entropy as the consistency loss function, a consistency loss weight of 10 and confidence thresholding French et al. [2018b] with a threshold of 0.97.

The ground truth decision boundary was derived from a hand-drawn 512×512 pixel image. The distance map shown in Figure 5.4(c) was computed using `scipy.ndimage.morphology.distance_transform_edt`, with distances negated for regions assigned to class 0. Each pixel in the distance map therefore has a signed distance to the ground truth class boundary. This distance map was used to generate the countours seen as lines in Figure 5.4(c) and used to support the constrained consistency regularization experiment illustrated in Figure 5.4(d).

The constrained consistency regularization experiment described in Section 5.3.1 required that a sample x should be perturbed to $\hat{x}$ such that they are at the same — or similar — distance to the ground truth decision boundary. This was achieved by drawing isotropic perturbations from a normal distrubtion $\hat{x} = x + h$ where $h \sim \mathcal{N}(0, 0.117)$ ($0.117 \approx 30$ pixels in the source image), determining the distances $m(x)$ and $m(\hat{x})$ from x and $\hat{x}$ to the ground truth boundary (using a pre-computed distance map) and discarding the perturbation – by masking consistency loss for x to 0 – if $|m(\hat{x}) - m(x)| > 0.016$ ($0.016 \approx 4$ pixels in the source image).

B Appendix: By-catch quantification

B.1 Results on validation set

After optimizing the discard counter hyper-parameters (see Section 7.8.5) the predictions are compared to the ground truths and the evaluation results are presented in the following confusion matrices and summary tables.

B.1.1 Validation: Vessel A

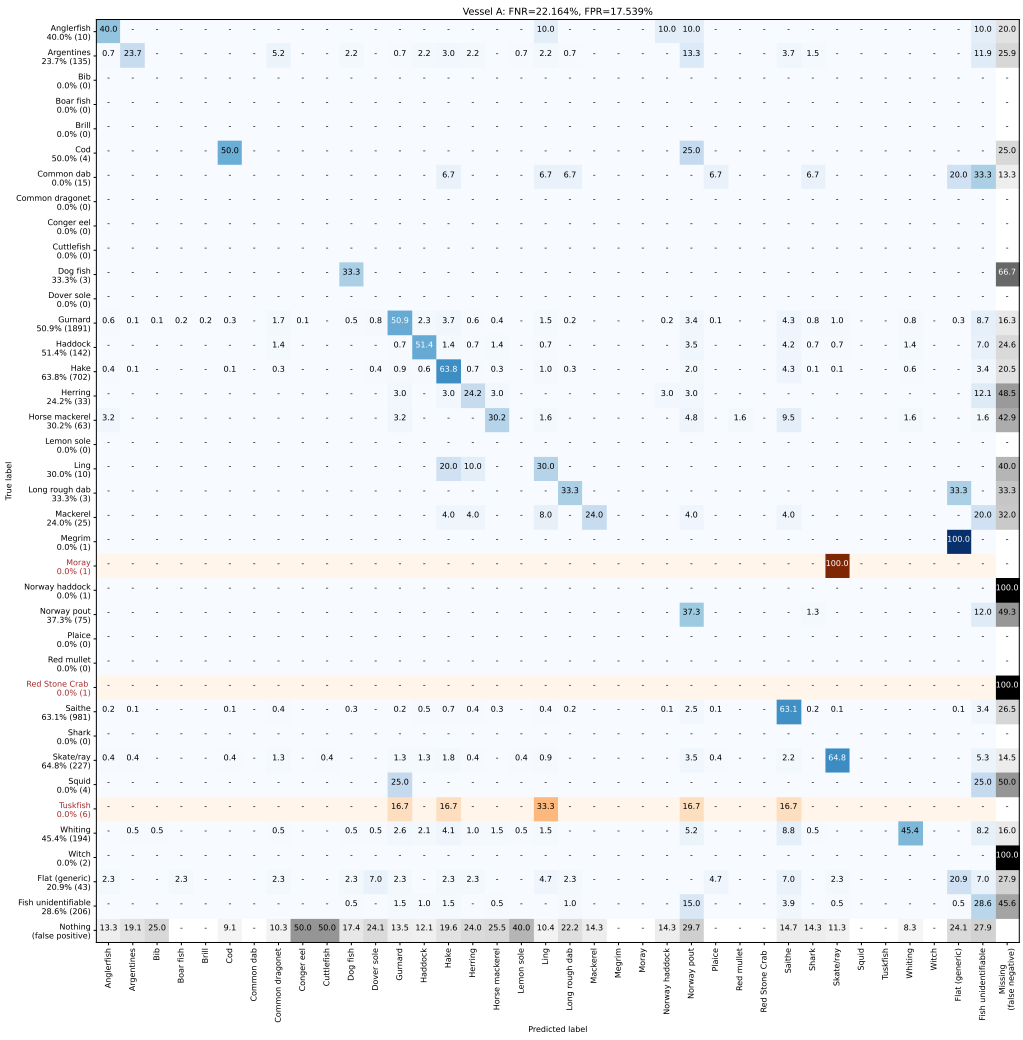


Figure B.1: Discard counter evaluation for vessel A presented as an extended confusion matrix. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions (non-zero values along the diagonal). The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from detection and tracking errors.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	10	30	36.76%	26.14%	41.67%	24.84%	33.33%	13.33%	40.00%	20.00%
Argentines	127	135	47	26.81%	67.22%	17.71%	25.56%	34.81%	68.09%	23.70%	35.16%
Bib	428	0	4	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Boar fish	1002	0	5	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Brill	22	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	4	11	35.00%	10.00%	50.00%	16.67%	36.36%	18.18%	50.00%	26.67%
Common dab	46	15	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	203	0	58	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Conger eel	1	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cuttlefish	4	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	3	23	25.00%	8.33%	33.33%	9.52%	13.04%	4.35%	33.33%	7.69%
Dover sole	360	0	29	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	1891	1144	52.84%	77.64%	42.53%	52.86%	60.50%	84.09%	50.87%	63.39%
Haddock	5498	142	157	52.32%	40.57%	40.22%	37.26%	90.45%	46.50%	51.41%	48.83%
Hake	578	702	688	56.65%	33.05%	33.93%	33.29%	98.01%	65.12%	63.82%	64.46%
Herring	410	33	50	46.94%	17.84%	21.43%	16.87%	66.00%	16.00%	24.24%	19.28%
Horse mackerel	100	63	51	54.79%	30.03%	19.58%	23.47%	80.95%	37.25%	30.16%	33.33%
Lemon sole	162	0	5	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Ling	18	10	67	26.13%	5.64%	50.00%	7.59%	14.93%	4.48%	30.00%	7.79%
Long rough dab	34	3	18	19.44%	11.11%	50.00%	16.67%	16.67%	5.56%	33.33%	9.52%
Mackerel	32	25	7	35.83%	88.89%	30.83%	41.67%	28.00%	85.71%	24.00%	37.50%
Megrim	108	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Moray	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Norway haddock	4	1	7	0.00%	0.00%	0.00%	0.00%	14.29%	0.00%	0.00%	0.00%
Norway pout	239	75	300	19.68%	33.69%	39.77%	16.21%	25.00%	9.33%	37.33%	14.93%
Plaice	534	0	6	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red mullet	4	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red Stone Crab	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	981	918	65.65%	51.88%	59.18%	52.74%	93.58%	67.43%	63.10%	65.19%
Shark	30	0	28	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	227	194	58.20%	82.95%	46.03%	55.97%	85.46%	75.77%	64.76%	69.83%
Squid	74	4	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Tuskfish	0	6	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	194	121	48.73%	75.40%	34.30%	41.33%	62.37%	72.73%	45.36%	55.87%
Witch	2	2	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Flat (generic)	538	43	29	52.31%	42.67%	10.65%	13.87%	67.44%	31.03%	20.93%	25.00%
Fish unidentifiable	7767	206	505	41.05%	10.47%	25.48%	14.41%	40.79%	11.68%	28.64%	16.60%
Any species	–	4778	4510	89.28%	45.31%	41.23%	43.08%	94.39%	55.63%	52.51%	54.03%
Average	–	4778	4510	20.38%	23.78%	24.87%	13.54%	26.00%	23.89%	27.50%	16.79%

Table B.1: Discard counter performance summary for vessel A. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.1.2 Validation: Vessel B

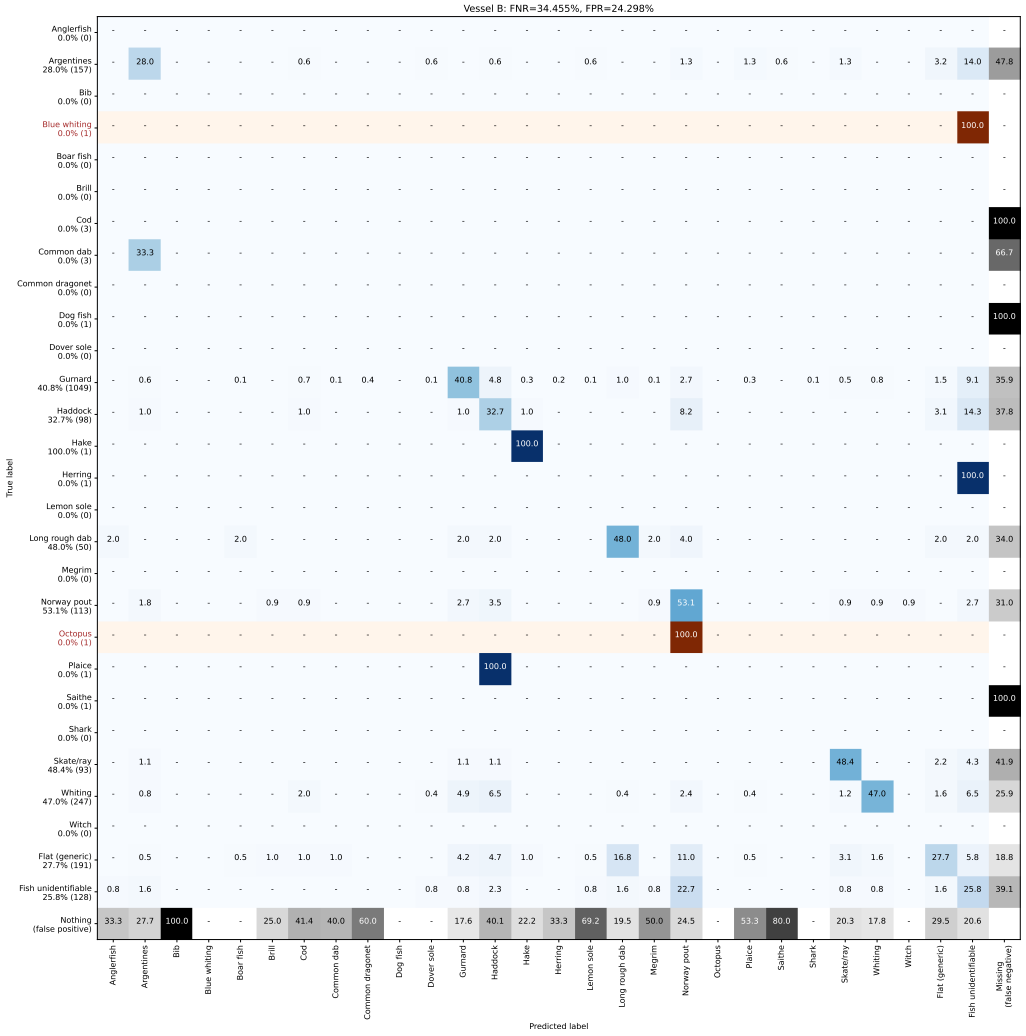


Figure B.2: Discard counter evaluation for vessel B presented as an extended confusion matrix. Please see the caption of Figure B.1 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Argentines	127	157	83	58.30%	32.88%	43.12%	29.02%	52.87%	53.01%	28.03%	36.67%
Bib	428	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Blue whiting	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Boar fish	1002	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Brill	22	0	4	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	3	29	6.67%	0.00%	0.00%	0.00%	10.34%	0.00%	0.00%	0.00%
Common dab	46	3	5	8.33%	0.00%	0.00%	0.00%	60.00%	0.00%	0.00%	0.00%
Common dragonet	203	0	10	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Dover sole	360	0	4	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	1049	552	57.02%	79.01%	44.15%	54.40%	52.62%	77.54%	40.80%	53.47%
Haddock	5498	98	197	33.17%	23.90%	35.61%	21.67%	49.75%	16.24%	32.65%	21.69%
Hake	578	1	9	10.00%	10.00%	100.00%	13.33%	11.11%	11.11%	100.00%	20.00%
Herring	410	1	3	0.00%	0.00%	0.00%	0.00%	33.33%	0.00%	0.00%	0.00%
Lemon sole	162	0	13	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Long rough dab	34	50	87	24.50%	31.00%	48.00%	24.22%	57.47%	27.59%	48.00%	35.04%
Megrim	108	0	8	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway pout	239	113	208	54.59%	26.89%	46.76%	33.10%	54.33%	28.85%	53.10%	37.38%
Octopus	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	534	1	15	5.00%	0.00%	0.00%	0.00%	6.67%	0.00%	0.00%	0.00%
Saithe	1477	1	5	33.33%	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%	0.00%
Shark	30	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	93	79	41.65%	41.33%	73.20%	42.40%	84.95%	56.96%	48.39%	52.33%
Whiting	446	247	157	65.82%	61.48%	45.25%	50.04%	63.56%	73.89%	46.96%	57.43%
Witch	2	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Flat (generic)	538	191	122	48.96%	34.97%	29.33%	25.06%	63.87%	43.44%	27.75%	33.87%
Fish unidentifiable	7767	128	253	29.05%	16.26%	22.58%	13.30%	50.59%	13.04%	25.78%	17.32%
Any species	–	2139	1852	78.78%	44.73%	40.69%	41.78%	86.58%	45.14%	39.08%	41.89%
Average	–	2139	1852	17.01%	14.31%	27.11%	10.95%	23.98%	16.07%	25.08%	13.04%

Table B.2: Discard counter performance summary for vessel B. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.1.3 Validation: Vessel C

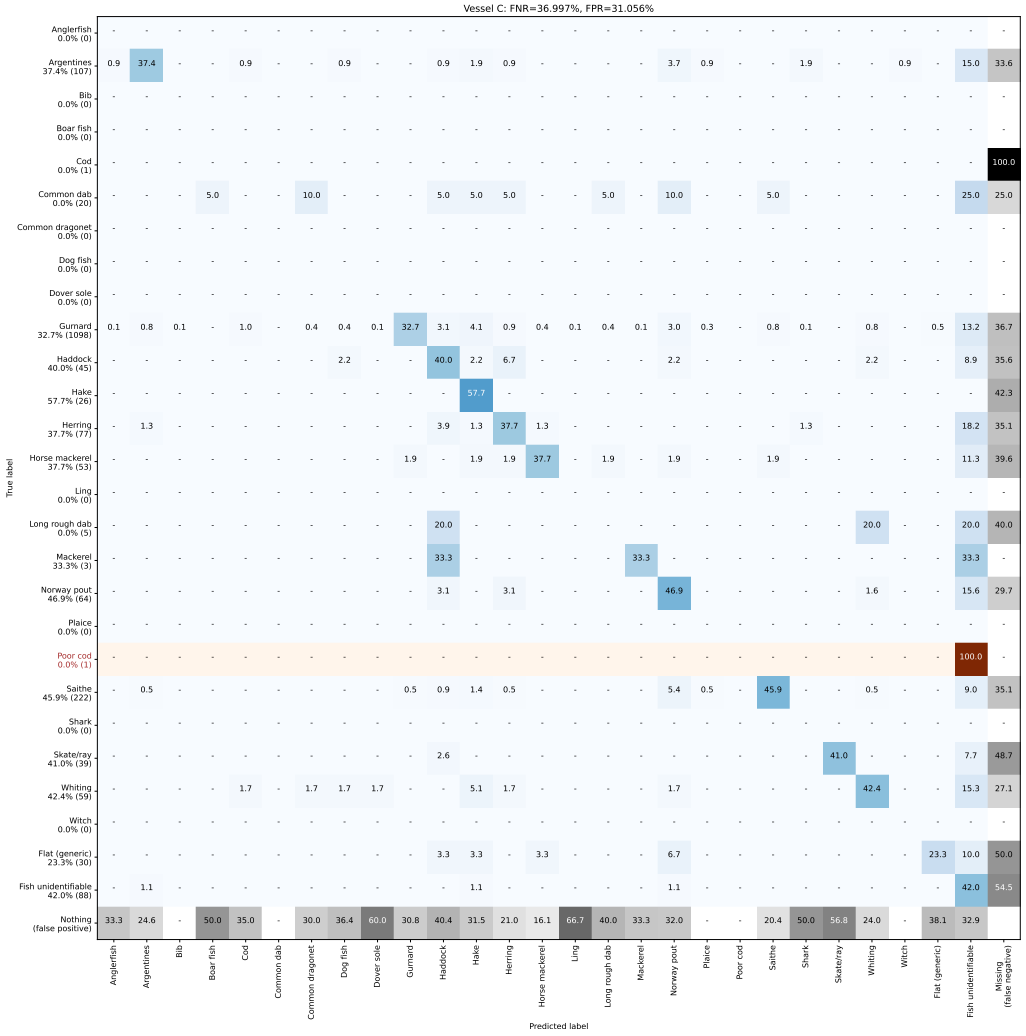


Figure B.3: Discard counter evaluation for vessel C presented as an extended confusion matrix. Please see the caption of Figure B.1 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Argentines	127	107	69	62.37%	54.42%	33.23%	40.45%	64.49%	57.97%	37.38%	45.45%
Bib	428	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Boar fish	1002	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	1	20	8.33%	0.00%	0.00%	0.00%	5.00%	0.00%	0.00%	0.00%
Common dab	46	20	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	203	0	10	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	0	11	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dover sole	360	0	5	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	1098	522	45.28%	73.32%	28.75%	38.97%	47.54%	68.77%	32.70%	44.32%
Haddock	5498	45	109	47.06%	20.22%	37.71%	25.58%	41.28%	16.51%	40.00%	23.38%
Hake	578	26	108	24.99%	13.46%	52.78%	14.71%	24.07%	13.89%	57.69%	22.39%
Herring	410	77	62	28.42%	27.40%	37.68%	21.57%	80.52%	46.77%	37.66%	41.73%
Horse mackerel	100	53	31	41.34%	26.67%	13.07%	17.54%	58.49%	64.52%	37.74%	47.62%
Ling	18	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Long rough dab	34	5	10	18.75%	0.00%	0.00%	0.00%	50.00%	0.00%	0.00%	0.00%
Mackerel	32	3	3	33.33%	25.00%	25.00%	16.67%	100.00%	33.33%	33.33%	33.33%
Norway pout	239	64	128	47.70%	28.07%	43.87%	31.27%	50.00%	23.44%	46.88%	31.25%
Plaice	534	0	5	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Poor cod	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	222	142	66.27%	54.77%	41.18%	44.85%	63.96%	71.83%	45.95%	56.04%
Shark	30	0	8	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	39	37	51.56%	51.28%	29.95%	32.04%	94.87%	43.24%	41.03%	42.11%
Whiting	446	59	50	49.51%	53.50%	39.38%	40.20%	84.75%	50.00%	42.37%	45.87%
Witch	2	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Flat (generic)	538	30	21	50.66%	48.21%	20.35%	27.50%	70.00%	33.33%	23.33%	27.45%
Fish unidentifiable	7767	88	410	21.32%	9.26%	46.26%	15.22%	21.46%	9.02%	42.05%	14.86%
Any species	–	1938	1771	74.34%	39.90%	34.90%	36.48%	91.38%	39.47%	36.07%	37.69%
Average	–	1938	1771	22.11%	19.42%	26.42%	13.58%	31.72%	21.31%	30.48%	17.62%

Table B.3: Discard counter performance summary for vessel C. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.1.4 Validation: Vessel D

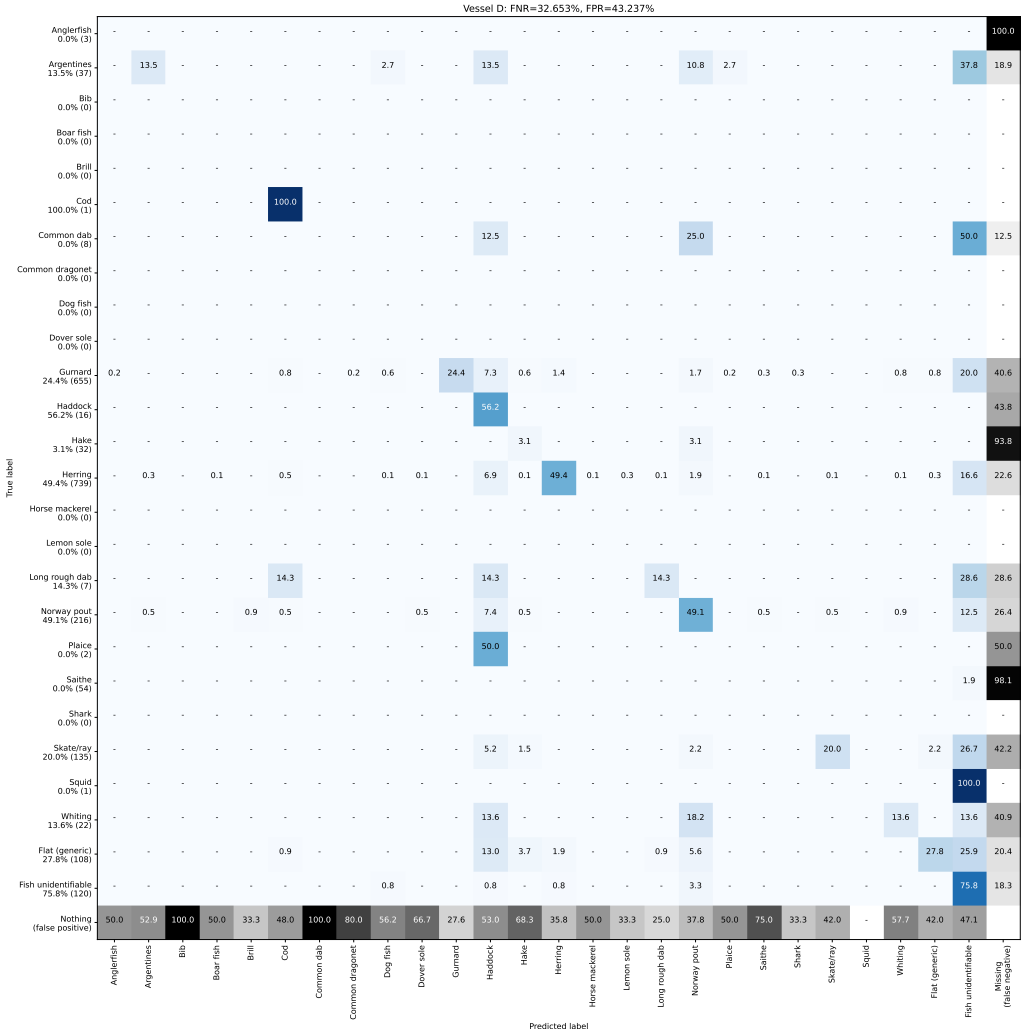


Figure B.4: Discard counter evaluation for vessel D presented as an extended confusion matrix. Please see the caption of Figure B.1 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	3	2	0.00%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Argentines	127	37	17	19.26%	18.75%	10.49%	8.57%	45.95%	29.41%	13.51%	18.52%
Bib	428	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Boar fish	1002	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Brill	22	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	1	25	5.00%	5.00%	100.00%	8.33%	4.00%	4.00%	100.00%	7.69%
Common dab	46	8	1	11.11%	0.00%	0.00%	0.00%	12.50%	0.00%	0.00%	0.00%
Common dragonet	203	0	5	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	0	16	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dover sole	360	0	6	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	6040	655	221	33.65%	65.07%	22.80%	31.53%	33.74%	72.40%	24.43%	36.53%
Haddock	5498	16	334	5.14%	3.18%	81.33%	5.94%	4.79%	2.69%	56.25%	5.14%
Hake	578	32	41	2.50%	5.00%	3.12%	1.11%	78.05%	2.44%	3.12%	2.74%
Herring	410	739	587	57.07%	58.37%	33.88%	40.56%	79.43%	62.18%	49.39%	55.05%
Horse mackerel	100	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Lemon sole	162	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Long rough dab	34	7	4	18.75%	16.67%	8.33%	7.14%	57.14%	25.00%	14.29%	18.18%
Norway pout	239	216	249	64.58%	46.50%	47.47%	44.31%	86.75%	42.57%	49.07%	45.59%
Plaice	534	2	4	16.67%	0.00%	0.00%	0.00%	50.00%	0.00%	0.00%	0.00%
Saithe	1477	54	16	11.51%	0.00%	0.00%	0.00%	29.63%	0.00%	0.00%	0.00%
Shark	30	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	135	50	54.08%	45.78%	24.08%	27.98%	37.04%	54.00%	20.00%	29.19%
Squid	74	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	22	26	53.96%	11.82%	42.86%	16.41%	84.62%	11.54%	13.64%	12.50%
Flat (generic)	538	108	69	57.93%	52.32%	31.05%	37.06%	63.89%	43.48%	27.78%	33.90%
Fish unidentifiable	7767	120	871	12.11%	9.20%	85.15%	15.31%	13.78%	10.45%	75.83%	18.37%
Any species	–	2156	2558	69.16%	27.72%	37.44%	31.32%	84.28%	31.24%	37.06%	33.90%
Average	–	2156	2558	16.28%	13.51%	28.86%	9.39%	28.77%	14.41%	26.31%	10.90%

Table B.4: Discard counter performance summary for vessel D. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.1.5 Validation: Vessel E

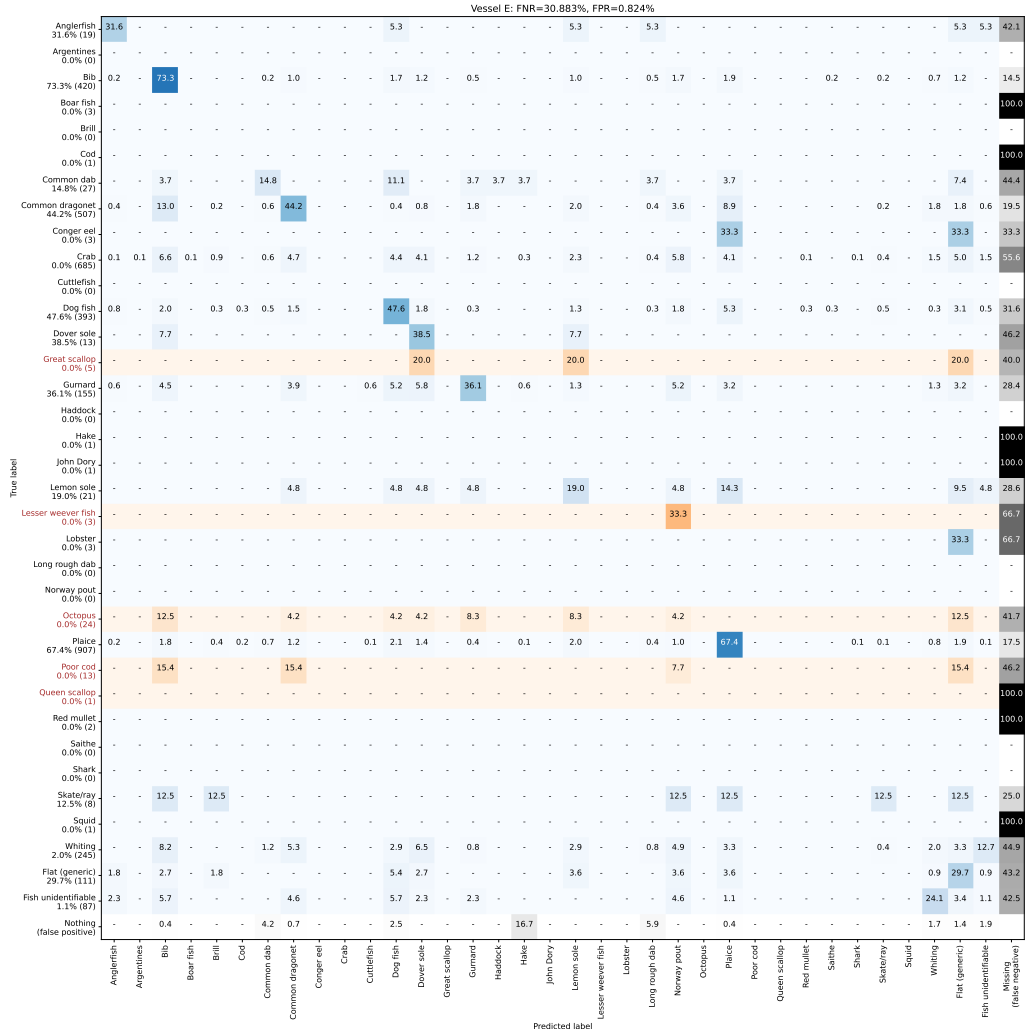


Figure B.5: Discard counter evaluation for vessel E presented as an extended confusion matrix. Please see the caption of Figure B.1 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	19	20	53.33%	27.14%	30.63%	23.21%	95.00%	30.00%	31.58%	30.77%
Argentines	127	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Bib	428	420	488	78.92%	46.78%	55.52%	49.60%	86.07%	63.11%	73.33%	67.84%
Boar fish	1002	3	1	33.33%	0.00%	0.00%	0.00%	33.33%	0.00%	0.00%	0.00%
Brill	22	0	15	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	1	3	0.00%	0.00%	0.00%	0.00%	33.33%	0.00%	0.00%	0.00%
Common dab	46	27	24	57.22%	13.89%	7.78%	9.84%	88.89%	16.67%	14.81%	15.69%
Common dragonet	203	507	306	65.40%	49.30%	37.76%	40.86%	60.36%	73.20%	44.18%	55.10%
Conger eel	1	3	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Crab	0	685	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	4	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	393	284	70.30%	56.04%	49.23%	49.99%	72.26%	65.85%	47.58%	55.24%
Dover sole	360	13	95	16.51%	3.30%	32.14%	5.75%	13.68%	5.26%	38.46%	9.26%
Great scallop	0	5	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	6040	155	88	53.44%	50.51%	25.44%	33.75%	56.77%	63.64%	36.13%	46.09%
Haddock	5498	0	1	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Hake	578	1	6	0.00%	0.00%	0.00%	0.00%	16.67%	0.00%	0.00%	0.00%
John Dory	15	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lemon sole	162	21	75	17.75%	3.70%	9.52%	4.68%	28.00%	5.33%	19.05%	8.33%
Lesser weever fish	0	3	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	0	3	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	34	0	17	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway pout	239	0	114	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Octopus	0	24	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	534	907	740	68.30%	74.98%	60.41%	64.16%	81.59%	82.57%	67.36%	74.20%
Poor cod	0	13	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Queen scallop	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red mullet	4	2	2	33.33%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%
Saithe	1477	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Shark	30	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	8	10	50.00%	8.33%	8.33%	6.25%	80.00%	10.00%	12.50%	11.11%
Squid	74	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	245	60	18.63%	7.78%	1.27%	2.05%	24.49%	8.33%	2.04%	3.28%
Flat (generic)	538	111	142	46.75%	23.01%	26.75%	20.95%	78.17%	23.24%	29.73%	26.09%
Fish unidentifiable	7767	87	52	38.31%	1.56%	1.67%	1.39%	59.77%	1.92%	1.15%	1.44%
Any species	–	3659	2550	68.13%	52.53%	35.92%	42.36%	69.69%	56.67%	39.49%	46.55%
Average	–	3659	2550	20.04%	14.65%	12.83%	8.93%	28.81%	17.97%	15.48%	11.56%

Table B.5: Discard counter performance summary for vessel E. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.1.6 Validation: Vessel F

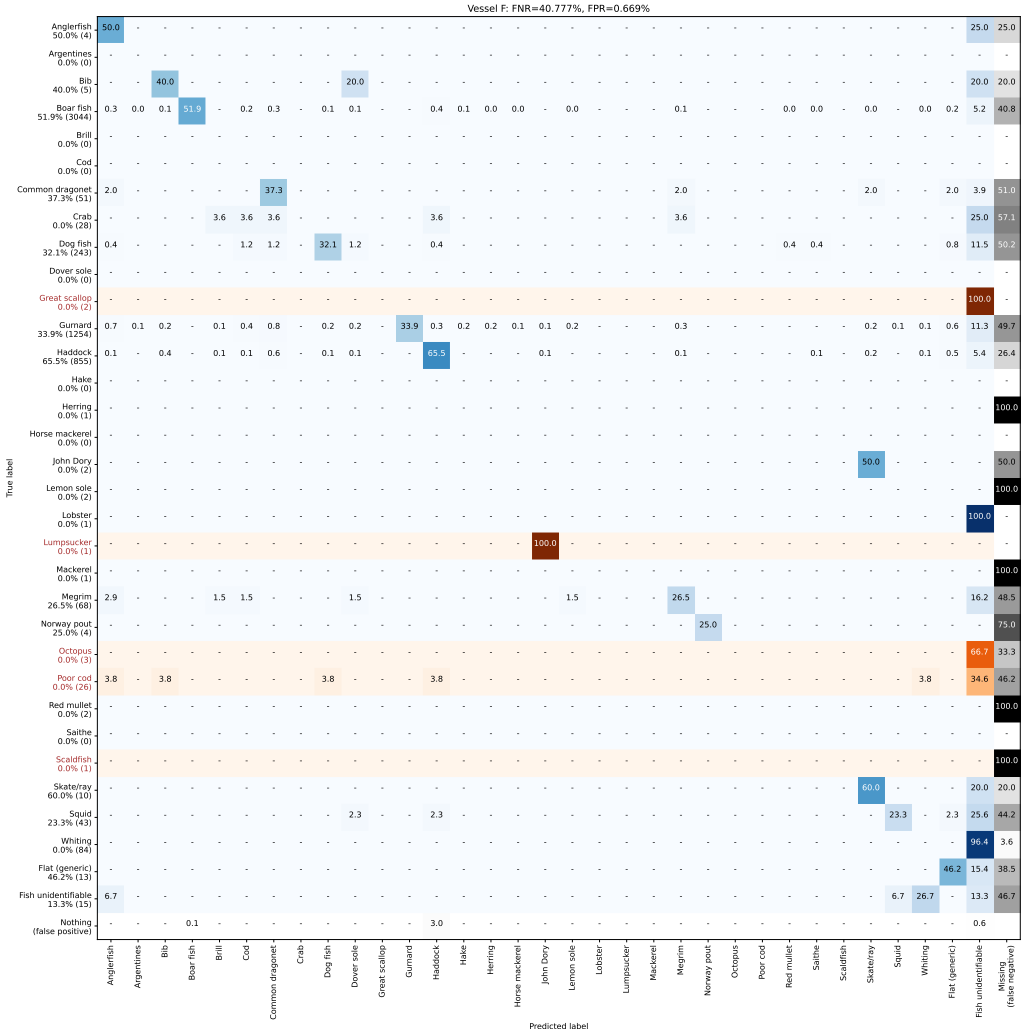


Figure B.6: Discard counter evaluation for vessel F presented as an extended confusion matrix. Please see the caption of Figure B.1 for a more detailed description.

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	4	27	11.67%	5.56%	33.33%	7.41%	14.81%	7.41%	50.00%	12.90%
Argentines	127	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Bib	428	5	11	10.00%	25.00%	62.50%	16.00%	45.45%	18.18%	40.00%	25.00%
Boar fish	1002	3044	1581	46.85%	99.88%	46.78%	61.86%	51.94%	99.87%	51.87%	68.28%
Brill	22	0	4	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	0	16	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Common dragonet	203	51	46	36.76%	30.71%	44.20%	21.82%	90.20%	41.30%	37.25%	39.18%
Crab	0	28	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Dog fish	376	243	87	59.50%	90.00%	53.92%	63.03%	35.80%	89.66%	32.10%	47.27%
Dover sole	360	0	13	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Great scallop	0	2	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	6040	1254	425	34.94%	100.00%	34.94%	50.21%	33.89%	100.00%	33.89%	50.63%
Haddock	5498	855	599	67.23%	88.26%	59.18%	69.82%	70.06%	93.49%	65.50%	77.03%
Hake	578	0	5	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Herring	410	1	3	50.00%	0.00%	0.00%	0.00%	33.33%	0.00%	0.00%	0.00%
Horse mackerel	100	0	2	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
John Dory	15	2	3	0.00%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Lemon sole	162	2	5	12.50%	0.00%	0.00%	0.00%	40.00%	0.00%	0.00%	0.00%
Lobster	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lumpsucker	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Mackerel	32	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Megrim	108	68	27	30.01%	61.94%	46.87%	36.26%	39.71%	66.67%	26.47%	37.89%
Norway pout	239	4	1	16.67%	100.00%	16.67%	22.22%	25.00%	100.00%	25.00%	40.00%
Octopus	0	3	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Poor cod	0	26	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red mullet	4	2	2	25.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%
Saithe	1477	0	3	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Scaldfish	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	431	10	14	51.94%	42.22%	55.00%	37.62%	71.43%	42.86%	60.00%	50.00%
Squid	74	43	12	39.85%	79.17%	27.35%	34.03%	27.91%	83.33%	23.26%	36.36%
Whiting	446	84	8	8.94%	0.00%	0.00%	0.00%	9.52%	0.00%	0.00%	0.00%
Flat (generic)	538	13	29	30.72%	14.62%	32.00%	16.38%	44.83%	20.69%	46.15%	28.57%
Fish unidentifiable	7767	15	511	2.59%	0.24%	13.33%	0.45%	2.94%	0.39%	13.33%	0.76%
Any species	–	5763	3436	60.60%	69.34%	42.24%	51.84%	59.62%	78.81%	46.99%	58.88%
Average	–	5763	3436	16.22%	29.50%	20.23%	13.25%	24.35%	30.55%	19.42%	15.57%

Table B.6: Discard counter performance summary for vessel F. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.1.7 Validation: All vessels

Species	# Samples in training data	True count	Predicted count	Video count accuracy	Video precision	Video recall	Video F1	Vessel count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	334	36	85	24.46%	12.57%	30.38%	12.14%	42.35%	14.12%	33.33%	19.83%
Argentines	127	436	219	35.45%	34.67%	28.18%	21.52%	50.23%	55.25%	27.75%	36.95%
Bib	428	425	506	38.01%	27.30%	56.79%	26.32%	83.99%	61.26%	72.94%	66.60%
Blue whiting	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Boar fish	1002	3047	1594	26.75%	49.94%	32.75%	27.06%	52.31%	99.06%	51.82%	68.05%
Brill	22	0	29	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cod	447	10	104	6.53%	1.74%	33.33%	2.78%	9.62%	2.88%	30.00%	5.26%
Common dab	46	73	30	27.70%	8.93%	3.89%	4.22%	41.10%	13.33%	5.48%	7.77%
Common dragonet	203	558	435	28.20%	21.95%	40.06%	17.35%	77.96%	55.86%	43.55%	48.94%
Conger eel	1	3	2	0.00%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Crab	0	713	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	4	0	4	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Dog fish	376	640	421	41.64%	44.59%	47.43%	34.06%	65.78%	63.18%	41.56%	50.14%
Dover sole	360	13	152	5.71%	1.14%	32.14%	1.99%	8.55%	3.29%	38.46%	6.06%
Great scallop	0	7	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	6040	6102	2952	45.77%	74.31%	32.58%	43.27%	48.38%	80.96%	39.17%	52.79%
Haddock	5498	1156	1397	41.74%	40.17%	52.69%	35.11%	82.75%	49.53%	59.86%	54.21%
Hake	578	762	857	15.56%	10.88%	39.72%	10.57%	88.91%	54.26%	61.02%	57.44%
Herring	410	851	705	38.16%	27.81%	26.29%	19.81%	82.84%	57.02%	47.24%	51.67%
Horse mackerel	100	116	86	31.20%	18.19%	16.79%	13.32%	74.14%	45.35%	33.62%	38.61%
John Dory	15	3	3	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%
Lemon sole	162	23	101	10.49%	1.67%	8.16%	2.11%	22.77%	3.96%	17.39%	6.45%
Lesser weever fish	0	3	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Ling	18	10	70	20.91%	4.52%	50.00%	6.07%	14.29%	4.29%	30.00%	7.50%
Lobster	0	4	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	34	65	136	13.32%	9.53%	22.10%	7.63%	47.79%	19.12%	40.00%	25.87%
Lumpsucker	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Mackerel	32	29	10	30.42%	63.33%	24.76%	27.08%	34.48%	70.00%	24.14%	35.90%
Megrim	108	69	35	16.37%	37.17%	39.06%	19.78%	50.72%	51.43%	26.09%	34.62%
Moray	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Norway haddock	4	1	7	0.00%	0.00%	0.00%	0.00%	14.29%	0.00%	0.00%	0.00%
Norway pout	239	472	1000	33.90%	28.56%	40.75%	23.84%	47.20%	22.50%	47.67%	30.57%
Octopus	0	28	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	534	910	770	34.23%	33.74%	49.42%	28.87%	84.62%	79.35%	67.14%	72.74%
Poor cod	0	40	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Queen scallop	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red mullet	4	4	5	25.00%	0.00%	0.00%	0.00%	80.00%	0.00%	0.00%	0.00%
Red Stone Crab	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	1477	1258	1086	34.26%	21.33%	36.49%	19.52%	86.33%	66.39%	57.31%	61.52%
Scaldfish	0	1	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Shark	30	0	42	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	431	512	384	50.82%	42.10%	37.70%	30.70%	75.00%	63.02%	47.27%	54.02%
Squid	74	49	12	26.57%	79.17%	18.23%	22.69%	24.49%	83.33%	20.41%	32.79%
Tuskfish	0	6	0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	446	851	422	36.47%	32.82%	22.59%	20.45%	49.59%	56.16%	27.85%	37.23%
Witch	2	2	2	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%
Flat (generic)	538	496	412	47.01%	33.47%	25.83%	23.12%	83.06%	33.50%	27.82%	30.40%
Fish unidentifiable	7767	644	2602	23.01%	6.49%	31.75%	8.13%	24.75%	8.57%	34.63%	13.74%
Any species	–	20433	16677	71.20%	49.44%	38.75%	42.28%	81.62%	53.94%	44.03%	48.48%
Average	–	20433	16677	16.87%	21.95%	19.55%	10.61%	37.81%	34.77%	23.41%	20.99%

Table B.7: Discard counter performance summary for all vessels. Rows in dark red indicate species that were not present in the training data, thus good performance cannot be expected. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.1.8 Validation: Video summaries

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel A	Video 0	Good	Easy	551	439	79.67%	55.13%	43.92%	48.89%
	Video 1	Good	Moderate	1570	1568	99.87%	54.15%	54.08%	54.11%
	Video 2	Fair	Moderate	2458	2339	95.16%	59.77%	56.88%	58.29%
	Video 3	Poor	Moderate	199	164	82.41%	12.20%	10.05%	11.02%
	Average			1194.5	1127.5	89.28%	45.31%	41.23%	43.08%
Vessel B	Video 0	Poor	Moderate	456	281	61.62%	48.75%	30.04%	37.18%
	Video 1	Fair	Moderate	528	514	97.35%	41.25%	40.15%	40.69%
	Video 2	Good	Hard	206	298	69.13%	36.91%	53.40%	43.65%
	Video 3	Good	Easy	243	216	88.89%	45.37%	40.33%	42.70%
	Video 4	Good	Moderate	706	543	76.91%	51.38%	39.52%	44.68%
	Average			427.8	370.4	78.78%	44.73%	40.69%	41.78%
Vessel C	Video 0	Poor	Moderate	363	267	73.55%	41.57%	30.58%	35.24%
	Video 1	Good	Hard	288	444	64.86%	21.62%	33.33%	26.23%
	Video 2	Good	Moderate	460	319	69.35%	52.66%	36.52%	43.13%
	Video 3	Fair	Moderate	827	741	89.60%	43.72%	39.18%	41.33%
	Average			484.5	442.75	74.34%	39.90%	34.90%	36.48%
Vessel D	Video 0	Fair	Moderate	585	932	62.77%	35.84%	57.09%	44.03%
	Video 1	Good	Hard	152	230	66.09%	28.26%	42.76%	34.03%
	Video 2	Good	Easy	69	117	58.97%	17.95%	30.43%	22.58%
	Video 3	Good	Moderate	975	791	81.13%	34.01%	27.59%	30.46%
	Video 4	Poor	Moderate	375	488	76.84%	22.54%	29.33%	25.49%
	Average			431.2	511.6	69.16%	27.72%	37.44%	31.32%
Vessel E	Video 0	Poor	Moderate	225	140	62.22%	47.14%	29.33%	36.16%
	Video 1	Poor	Moderate	252	107	42.46%	42.99%	18.25%	25.63%
	Video 2	Poor	Moderate	161	101	62.73%	47.52%	29.81%	36.64%
	Video 3	Fair	Moderate	265	223	84.15%	26.46%	22.26%	24.18%
	Video 4	Good	Hard	512	407	79.49%	61.67%	49.02%	54.62%
	Video 5	Good	Moderate	539	428	79.41%	65.19%	51.76%	57.70%
	Video 6	Good	Moderate-hard	456	342	75.00%	59.65%	44.74%	51.13%
	Video 7	Fair	Moderate	543	331	60.96%	59.52%	36.28%	45.08%
	Video 8	Fair	Moderate-hard	706	471	66.71%	62.63%	41.78%	50.13%
	Average			406.555556	283.333333	68.13%	52.53%	35.92%	42.36%
Vessel F	Video 0	Fair	Easy-moderate	2477	1579	63.75%	81.51%	51.96%	63.46%
	Video 1	Fair	Easy	657	427	64.99%	83.37%	54.19%	65.68%
	Video 2	Fair	Easy	1075	692	64.37%	82.23%	52.93%	64.40%
	Video 3	Good	Moderate-hard	796	249	31.28%	65.86%	20.60%	31.39%
	Video 4	Good	Hard	506	302	59.68%	66.89%	39.92%	50.00%
	Video 5	Good	Easy	36	18	50.00%	61.11%	30.56%	40.74%
	Video 6	Good	Easy	172	138	80.23%	78.26%	62.79%	69.68%
	Video 7	Fair	Easy	44	31	70.45%	35.48%	25.00%	29.33%
	Average			720.375	429.5	60.60%	69.34%	42.24%	51.84%

Table B.8: Discard counter performance per-video summary for vessels A-F.

B.2 Inter-observer variability: results of individual observers on test set

B.2.1 Inter-observer variability on test set: Vessel A

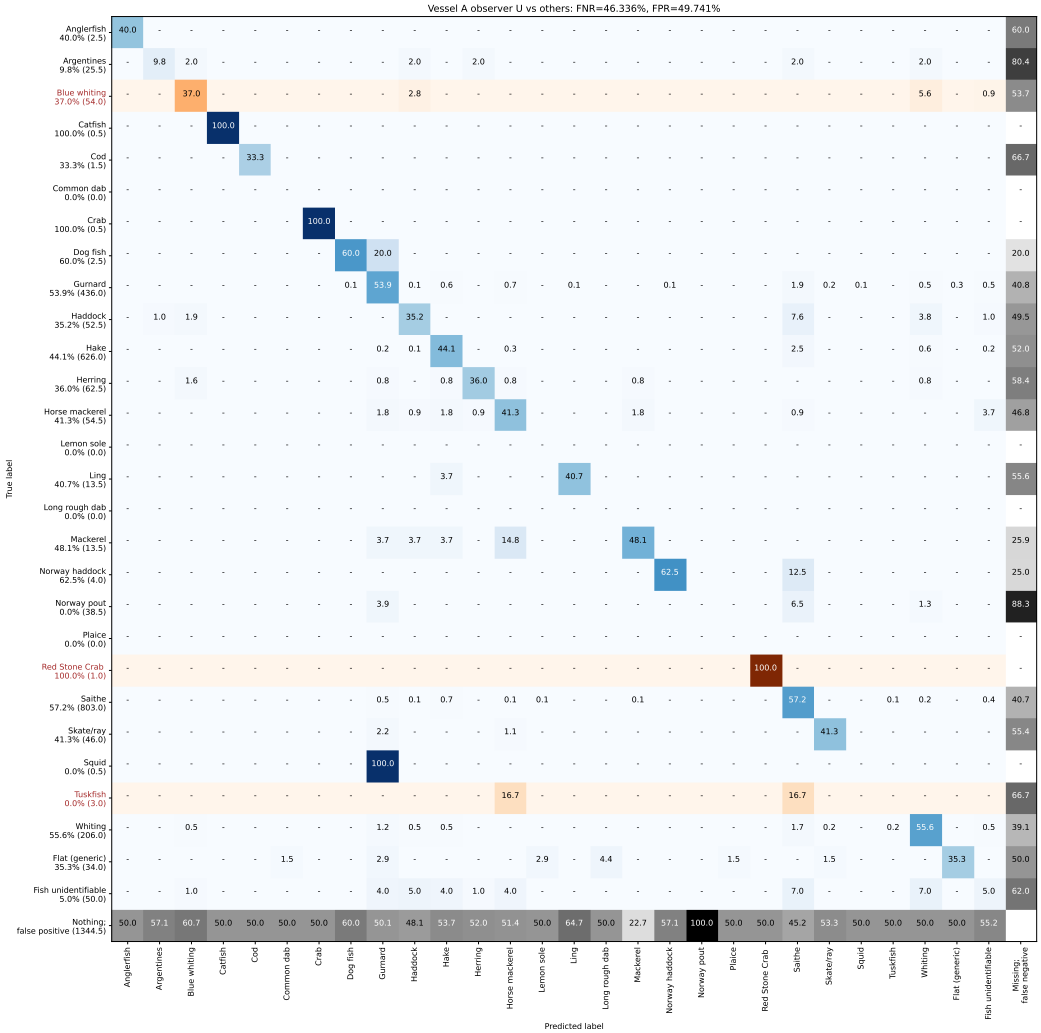


Figure B.7: Inter-observer evaluation for observer U, vessel A presented as an extended confusion matrix, contrasting the observations of observer U vs observers V and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. U count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	2.5	2.0	16.67%	50.00%	33.33%	22.22%	80.00%	50.00%	40.00%	44.44%
Argentines	25.5	7.0	23.02%	25.00%	5.95%	8.06%	27.45%	35.71%	9.80%	15.38%
Blue whiting	54.0	61.0	43.85%	32.79%	18.69%	17.47%	88.52%	32.79%	37.04%	34.78%
Catfish	0.5	1.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Cod	1.5	1.0	66.67%	50.00%	33.33%	40.00%	66.67%	50.00%	33.33%	40.00%
Common dab	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Crab	0.5	1.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Dog fish	2.5	5.0	50.00%	30.00%	60.00%	40.00%	50.00%	30.00%	60.00%	40.00%
Gurnard	436.0	503.0	80.46%	45.46%	49.43%	46.68%	86.68%	46.72%	53.90%	50.05%
Haddock	52.5	52.0	51.54%	23.97%	28.68%	22.28%	99.05%	35.58%	35.24%	35.41%
Hake	626.0	625.0	85.32%	34.18%	35.24%	34.53%	99.84%	44.16%	44.09%	44.12%
Herring	62.5	50.0	57.35%	34.32%	29.72%	27.68%	80.00%	45.00%	36.00%	40.00%
Horse mackerel	54.5	69.0	53.82%	15.14%	28.13%	15.84%	78.99%	32.61%	41.28%	36.44%
Lemon sole	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Ling	13.5	17.0	80.00%	32.00%	40.71%	35.26%	79.41%	32.35%	40.74%	36.07%
Long rough dab	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Mackerel	13.5	11.0	37.92%	55.00%	25.98%	26.22%	81.48%	59.09%	48.15%	53.06%
Norway haddock	4.0	7.0	57.14%	35.71%	62.50%	45.45%	57.14%	35.71%	62.50%	45.45%
Norway pout	38.5	4.0	7.03%	0.00%	0.00%	0.00%	10.39%	0.00%	0.00%	0.00%
Plaice	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red Stone Crab	1.0	2.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Saithe	803.0	909.0	67.19%	48.31%	47.40%	42.98%	88.34%	50.50%	57.16%	53.62%
Skate/ray	46.0	45.0	48.78%	46.23%	54.49%	41.85%	97.83%	42.22%	41.30%	41.76%
Squid	0.5	1.0	50.00%	0.00%	0.00%	0.00%	50.00%	0.00%	0.00%	0.00%
Tuskfish	3.0	2.0	66.67%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Whiting	206.0	264.0	71.78%	41.77%	58.45%	48.56%	78.03%	43.37%	55.58%	48.72%
Flat (generic)	34.0	27.0	32.28%	46.74%	17.20%	18.64%	79.41%	44.44%	35.29%	39.34%
Fish unidentifiable	50.0	29.0	43.29%	6.12%	4.05%	4.52%	58.00%	8.62%	5.00%	6.33%
Any species	2531.5	2703.0	92.03%	41.51%	43.58%	42.46%	93.66%	45.26%	48.33%	46.75%
Average	2531.5	2703.0	44.31%	28.67%	38.89%	26.37%	59.07%	31.03%	43.18%	32.32%

Table B.9: Inter-observer evaluation summary for observer U, vessel A, contrasting the observations of observer U vs observers V and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

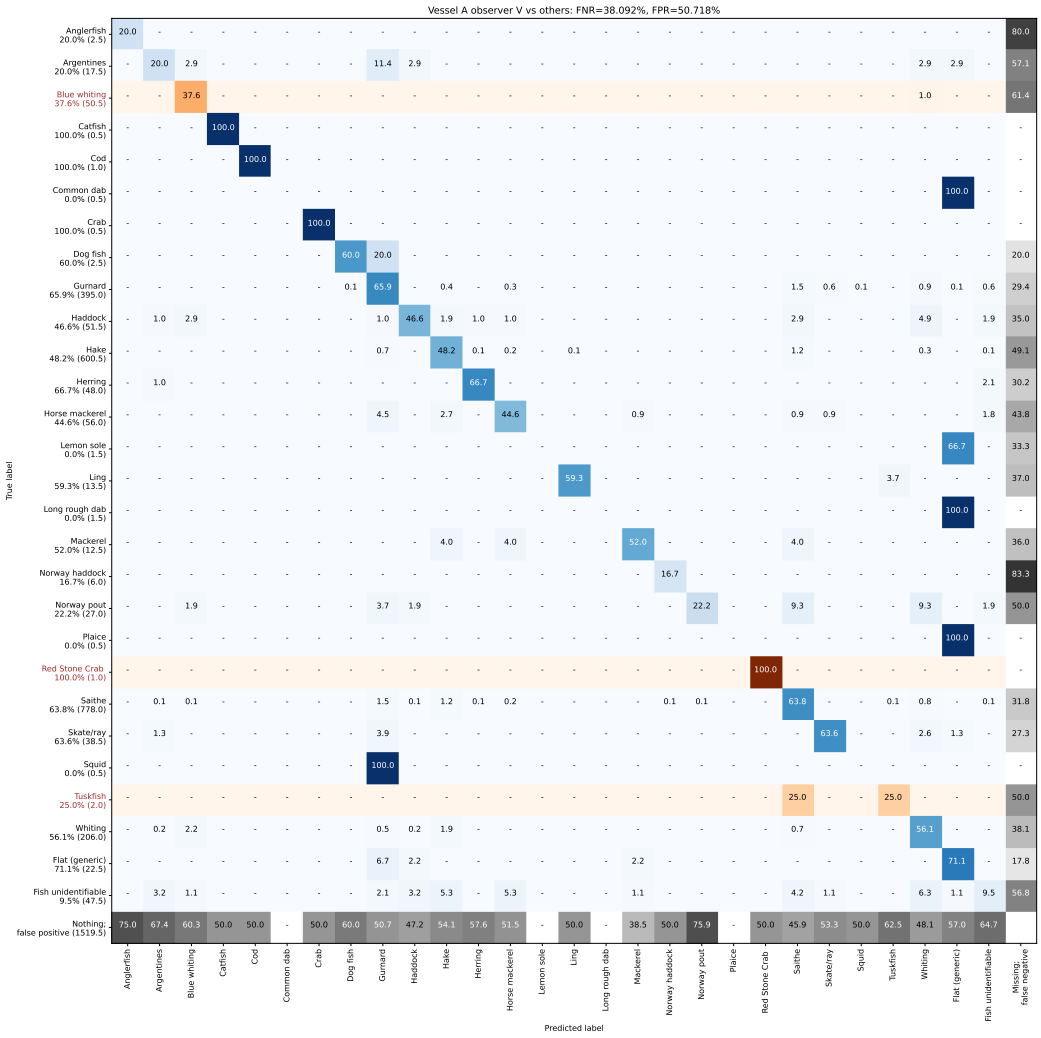


Figure B.8: Inter-observer evaluation for observer V, vessel A presented as an extended confusion matrix, contrasting the observations of observer V vs observers U and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. V count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	2.5	2.0	16.67%	25.00%	50.00%	22.22%	80.00%	25.00%	20.00%	22.22%
Argentines	17.5	23.0	27.22%	22.22%	21.25%	10.98%	76.09%	15.22%	20.00%	17.28%
Blue whiting	50.5	68.0	37.69%	14.18%	37.62%	16.17%	74.26%	27.94%	37.62%	32.07%
Catfish	0.5	1.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Cod	1.0	2.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Common dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Crab	0.5	1.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Dog fish	2.5	5.0	50.00%	30.00%	60.00%	40.00%	50.00%	30.00%	60.00%	40.00%
Gurnard	395.0	585.0	67.90%	37.34%	64.18%	43.80%	67.52%	44.53%	65.95%	53.16%
Haddock	51.5	54.0	50.53%	42.75%	30.66%	27.91%	95.37%	44.44%	46.60%	45.50%
Hake	600.5	676.0	73.35%	39.46%	53.62%	44.45%	88.83%	42.83%	48.21%	45.36%
Herring	48.0	79.0	60.86%	28.36%	29.39%	23.56%	60.76%	40.51%	66.67%	50.39%
Horse mackerel	56.0	66.0	52.05%	25.83%	19.51%	17.25%	84.85%	37.88%	44.64%	40.98%
Lemon sole	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Ling	13.5	17.0	58.17%	47.50%	54.57%	44.35%	79.41%	47.06%	59.26%	52.46%
Long rough dab	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Mackerel	12.5	13.0	33.75%	28.33%	40.20%	24.55%	96.15%	50.00%	52.00%	50.98%
Norway haddock	6.0	3.0	50.00%	33.33%	16.67%	22.22%	50.00%	33.33%	16.67%	22.22%
Norway pout	27.0	27.0	27.73%	23.08%	9.09%	10.17%	100.00%	22.22%	22.22%	22.22%
Plaice	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red Stone Crab	1.0	2.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Saithe	778.0	959.0	59.69%	43.22%	71.25%	51.77%	81.13%	51.77%	63.82%	57.17%
Skate/ray	38.5	60.0	54.47%	46.52%	88.32%	60.71%	64.17%	40.83%	63.64%	49.75%
Squid	0.5	1.0	50.00%	0.00%	0.00%	0.00%	50.00%	0.00%	0.00%	0.00%
Tuskfish	2.0	4.0	50.00%	12.50%	25.00%	16.67%	50.00%	12.50%	25.00%	16.67%
Whiting	206.0	264.0	69.52%	40.41%	58.92%	46.73%	78.03%	43.75%	56.07%	49.15%
Flat (generic)	22.5	50.0	34.74%	22.18%	64.24%	30.18%	45.00%	32.00%	71.11%	44.14%
Fish unidentifiable	47.5	34.0	62.05%	12.96%	6.42%	8.33%	71.58%	13.24%	9.47%	11.04%
Any species	2385.0	2996.0	72.95%	40.71%	57.59%	46.81%	79.61%	44.64%	56.08%	49.71%
Average	2385.0	2996.0	42.37%	32.30%	42.89%	29.60%	58.68%	35.63%	44.61%	35.34%

Table B.10: Inter-observer evaluation summary for observer V, vessel A, contrasting the observations of observer V vs observers U and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

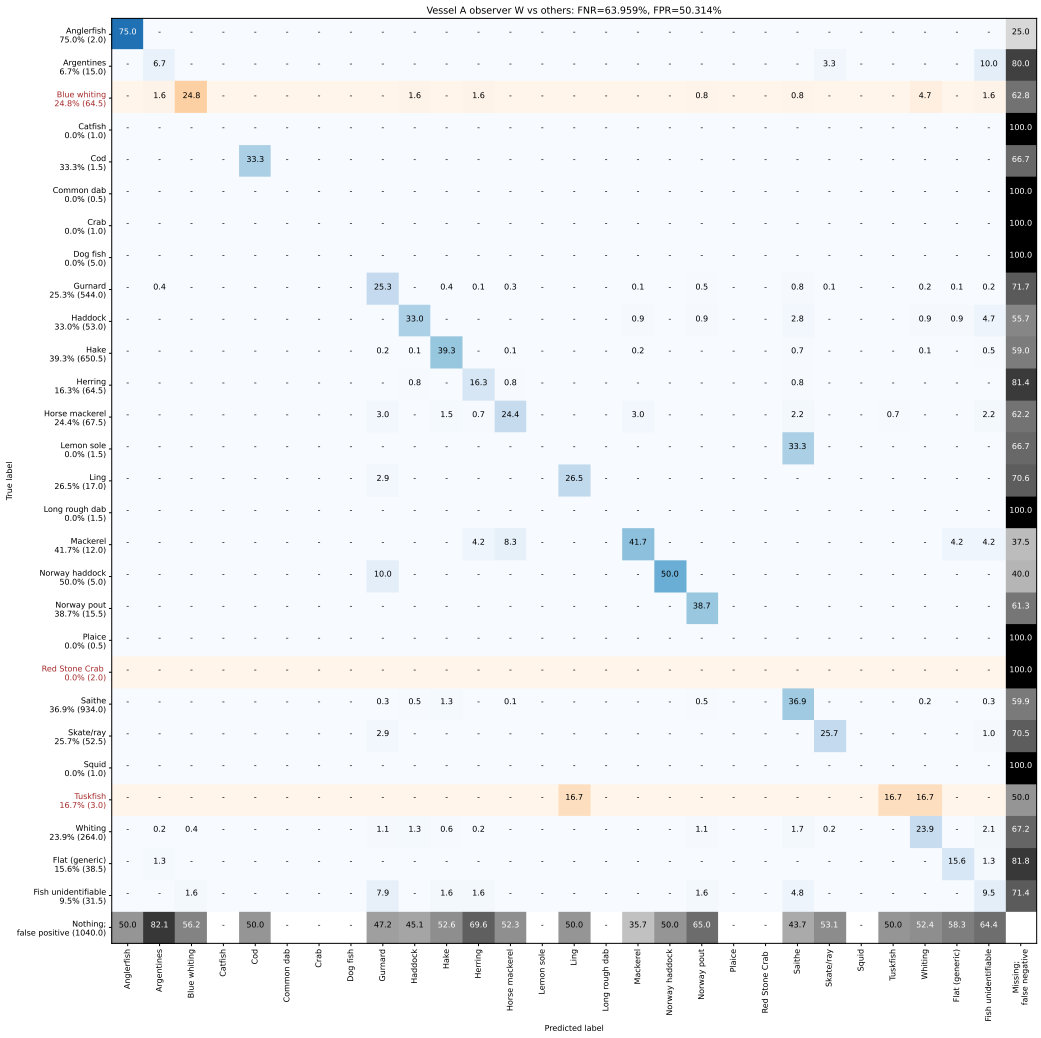


Figure B.9: Inter-observer evaluation for observer W, vessel A presented as an extended confusion matrix, contrasting the observations of observer W vs observers U and V. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. W count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	2.0	3.0	33.33%	50.00%	66.67%	44.44%	66.67%	50.00%	75.00%	60.00%
Argentines	15.0	28.0	15.38%	5.73%	12.56%	4.36%	53.57%	3.57%	6.67%	4.65%
Blue whiting	64.5	40.0	31.25%	40.00%	12.50%	15.38%	62.02%	40.00%	24.81%	30.62%
Catfish	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cod	1.5	1.0	66.67%	50.00%	33.33%	40.00%	66.67%	50.00%	33.33%	40.00%
Common dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Crab	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Dog fish	5.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	544.0	287.0	50.35%	43.51%	21.29%	26.15%	52.76%	47.91%	25.28%	33.09%
Haddock	53.0	51.0	49.10%	25.45%	18.40%	18.41%	96.23%	34.31%	33.02%	33.65%
Hake	650.5	576.0	67.94%	46.49%	31.29%	35.75%	88.55%	44.36%	39.28%	41.66%
Herring	64.5	46.0	49.95%	18.70%	10.09%	12.91%	71.32%	22.83%	16.28%	19.00%
Horse mackerel	67.5	43.0	47.02%	25.76%	10.33%	12.34%	63.70%	38.37%	24.44%	29.86%
Lemon sole	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Ling	17.0	10.0	56.70%	45.00%	31.21%	33.62%	58.82%	45.00%	26.47%	33.33%
Long rough dab	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Mackerel	12.0	14.0	34.17%	23.81%	16.67%	14.71%	85.71%	35.71%	41.67%	38.46%
Norway haddock	5.0	5.0	100.00%	50.00%	50.00%	50.00%	100.00%	50.00%	50.00%	50.00%
Norway pout	15.5	50.0	27.92%	4.84%	40.00%	7.79%	31.00%	12.00%	38.71%	18.32%
Plaice	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red Stone Crab	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	934.0	647.0	58.97%	38.72%	31.85%	33.36%	69.27%	53.25%	36.88%	43.58%
Skate/ray	52.5	32.0	37.05%	45.16%	31.65%	27.12%	60.95%	42.19%	25.71%	31.95%
Squid	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Tuskfish	3.0	2.0	66.67%	25.00%	16.67%	20.00%	66.67%	25.00%	16.67%	20.00%
Whiting	264.0	148.0	45.91%	36.99%	17.88%	22.66%	56.06%	42.57%	23.86%	30.58%
Flat (generic)	38.5	18.0	38.43%	30.00%	20.37%	18.74%	46.75%	33.33%	15.58%	21.24%
Fish unidentifiable	31.5	66.0	48.51%	2.45%	9.23%	3.86%	47.73%	4.55%	9.52%	6.15%
Any species	2849.5	2067.0	68.15%	39.82%	27.50%	31.82%	72.54%	43.78%	31.76%	36.81%
Average	2849.5	2067.0	33.05%	31.98%	17.21%	15.77%	44.44%	35.52%	20.11%	20.93%

Table B.11: Inter-observer evaluation summary for observer W, vessel A, contrasting the observations of observer W vs observers U and V. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.2.2 Inter-observer variability on test set: Vessel B

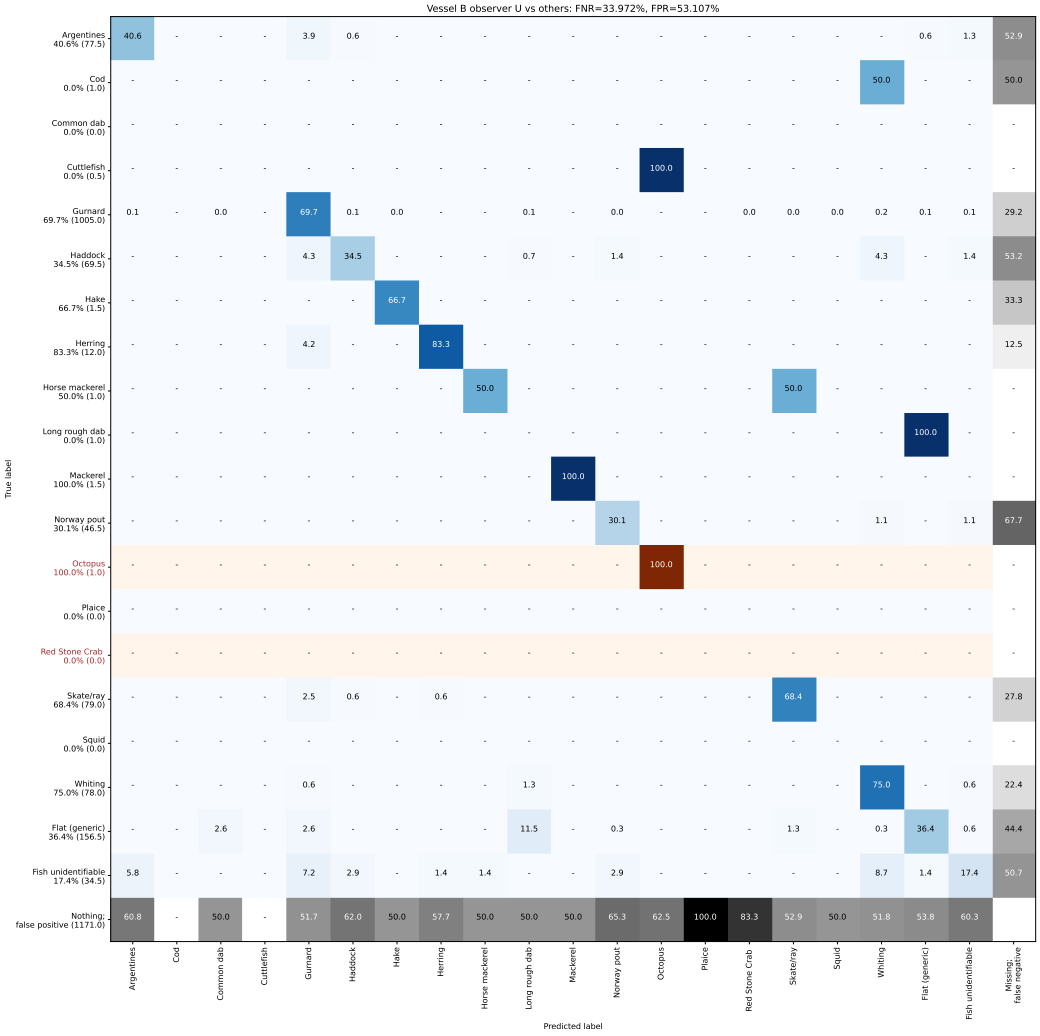


Figure B.10: Inter-observer evaluation for observer U, vessel B presented as an extended confusion matrix, contrasting the observations of observer U vs observers V and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. U count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	77.5	88.0	64.54%	37.86%	65.29%	45.80%	88.07%	35.80%	40.65%	38.07%
Cod	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	0.0	9.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Cuttlefish	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	1005.0	1481.0	66.05%	46.51%	70.95%	55.20%	67.86%	47.27%	69.65%	56.32%
Haddock	69.5	71.0	73.42%	34.24%	38.54%	34.31%	97.89%	33.80%	34.53%	34.16%
Hake	1.5	3.0	25.00%	37.50%	66.67%	35.56%	50.00%	33.33%	66.67%	44.44%
Herring	12.0	26.0	55.07%	36.23%	70.00%	47.27%	46.15%	38.46%	83.33%	52.63%
Horse mackerel	1.0	2.0	12.50%	25.00%	50.00%	20.00%	50.00%	25.00%	50.00%	33.33%
Long rough dab	1.0	41.0	0.78%	0.00%	0.00%	0.00%	2.44%	0.00%	0.00%	0.00%
Mackerel	1.5	3.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Norway pout	46.5	49.0	94.90%	28.57%	30.11%	29.32%	94.90%	28.57%	30.11%	29.32%
Octopus	1.0	4.0	16.67%	16.67%	100.00%	25.00%	25.00%	25.00%	100.00%	40.00%
Plaice	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Red Stone Crab	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	79.0	121.0	56.68%	44.22%	82.78%	56.76%	65.29%	44.63%	68.35%	54.00%
Squid	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Whiting	78.0	142.0	62.26%	36.77%	65.00%	44.59%	54.93%	41.20%	75.00%	53.18%
Flat (generic)	156.5	131.0	67.62%	37.06%	33.86%	33.76%	83.71%	43.51%	36.42%	39.65%
Fish unidentifiable	34.5	29.0	60.91%	18.89%	14.23%	15.36%	84.06%	20.69%	17.39%	18.90%
Any species	1566.0	2205.0	69.42%	42.79%	66.27%	51.24%	71.02%	43.49%	61.24%	50.86%
Average	1566.0	2205.0	35.32%	24.97%	49.21%	25.48%	43.01%	25.96%	48.26%	28.03%

Table B.12: Inter-observer evaluation summary for observer U, vessel B, contrasting the observations of observer U vs observers V and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Species	Other obs. count	Obs. V count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	83.5	76.0	81.67%	46.83%	52.78%	47.76%	91.02%	44.08%	40.12%	42.01%
Cod	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Common dab	4.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	1129.5	1232.0	79.18%	48.09%	50.48%	48.00%	91.68%	48.90%	53.34%	51.03%
Haddock	65.5	79.0	77.76%	35.12%	43.93%	38.39%	82.91%	39.24%	47.33%	42.91%
Hake	1.5	3.0	50.00%	33.33%	75.00%	38.89%	50.00%	33.33%	66.67%	44.44%
Herring	15.0	20.0	71.57%	48.53%	68.27%	56.67%	75.00%	47.50%	63.33%	54.29%
Horse mackerel	1.0	2.0	50.00%	25.00%	50.00%	25.00%	50.00%	25.00%	50.00%	33.33%
Long rough dab	20.5	2.0	3.12%	0.00%	0.00%	0.00%	9.76%	0.00%	0.00%	0.00%
Mackerel	2.0	2.0	25.00%	50.00%	50.00%	33.33%	100.00%	50.00%	50.00%	50.00%
Norway pout	42.0	58.0	72.41%	43.97%	60.71%	51.00%	72.41%	43.97%	60.71%	51.00%
Octopus	3.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Plaice	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red Stone Crab	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	99.5	80.0	55.91%	48.51%	34.05%	35.14%	80.40%	48.12%	38.69%	42.90%
Squid	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	96.0	106.0	51.49%	26.97%	50.67%	32.63%	90.57%	42.45%	46.88%	44.55%
Flat (generic)	119.5	205.0	43.57%	19.13%	37.30%	25.05%	58.29%	33.90%	58.16%	42.84%
Fish unidentifiable	38.5	21.0	63.64%	10.69%	9.11%	8.67%	54.55%	11.90%	6.49%	8.40%
Any species	1724.0	1889.0	80.59%	44.93%	48.39%	45.77%	91.27%	45.53%	49.88%	47.61%
Average	1724.0	1889.0	36.27%	29.08%	32.35%	22.03%	45.33%	31.23%	32.32%	25.38%

Table B.13: Inter-observer evaluation summary for observer V, vessel B, contrasting the observations of observer V vs observers U and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

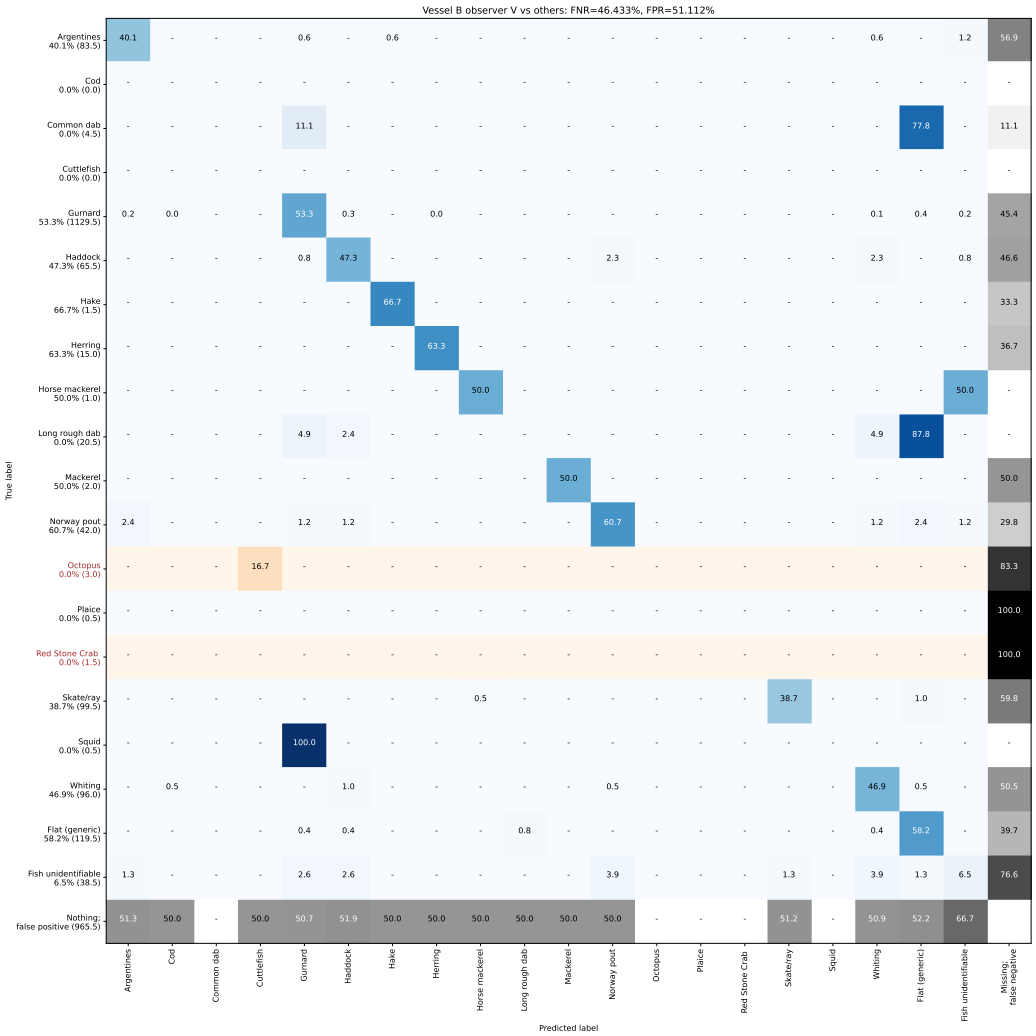


Figure B.11: Inter-observer evaluation for observer V, vessel B presented as an extended confusion matrix, contrasting the observations of observer V vs observers U and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

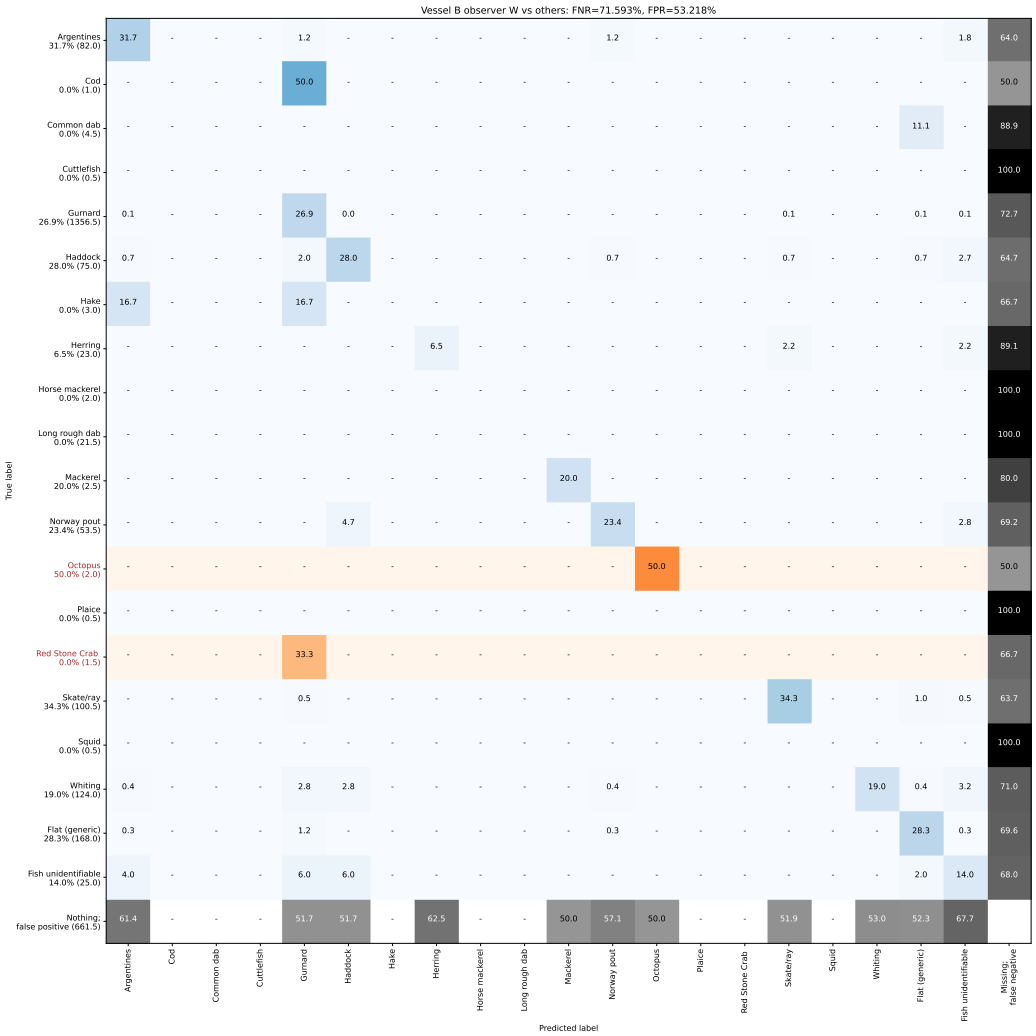


Figure B.12: Inter-observer evaluation for observer W, vessel B presented as an extended confusion matrix, contrasting the observations of observer W vs observers U and V. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. W count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	82.0	79.0	54.36%	35.61%	20.42%	21.13%	96.34%	32.91%	31.71%	32.30%
Cod	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	4.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Cuttlefish	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	1356.5	778.0	62.52%	43.97%	29.95%	33.06%	57.35%	46.85%	26.87%	34.15%
Haddock	75.0	60.0	63.56%	26.55%	23.25%	23.96%	80.00%	35.00%	28.00%	31.11%
Hake	3.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Herring	23.0	4.0	24.17%	41.67%	10.83%	16.85%	17.39%	37.50%	6.52%	11.11%
Horse mackerel	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	21.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Mackerel	2.5	1.0	25.00%	50.00%	50.00%	33.33%	40.00%	50.00%	20.00%	28.57%
Norway pout	53.5	35.0	65.42%	35.71%	23.36%	28.25%	65.42%	35.71%	23.36%	28.25%
Octopus	2.0	2.0	37.50%	50.00%	33.33%	28.57%	100.00%	50.00%	50.00%	50.00%
Plaice	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red Stone Crab	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	100.5	78.0	55.72%	43.89%	44.07%	38.70%	77.61%	44.23%	34.33%	38.66%
Squid	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	124.0	50.0	36.12%	46.15%	17.01%	22.45%	40.32%	47.00%	18.95%	27.01%
Flat (generic)	168.0	108.0	65.99%	33.20%	26.20%	27.85%	64.29%	43.98%	28.27%	34.42%
Fish unidentifiable	25.0	48.0	55.83%	4.26%	9.56%	5.32%	52.08%	7.29%	14.00%	9.59%
Any species	2047.0	1243.0	60.20%	42.84%	27.25%	31.12%	60.72%	43.12%	26.18%	32.58%
Average	2047.0	1243.0	27.31%	37.37%	14.40%	13.97%	34.54%	39.13%	14.10%	16.26%

Table B.14: Inter-observer evaluation summary for observer W, vessel B, contrasting the observations of observer W vs observers U and V. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.2.3 Inter-observer variability on test set: Vessel C

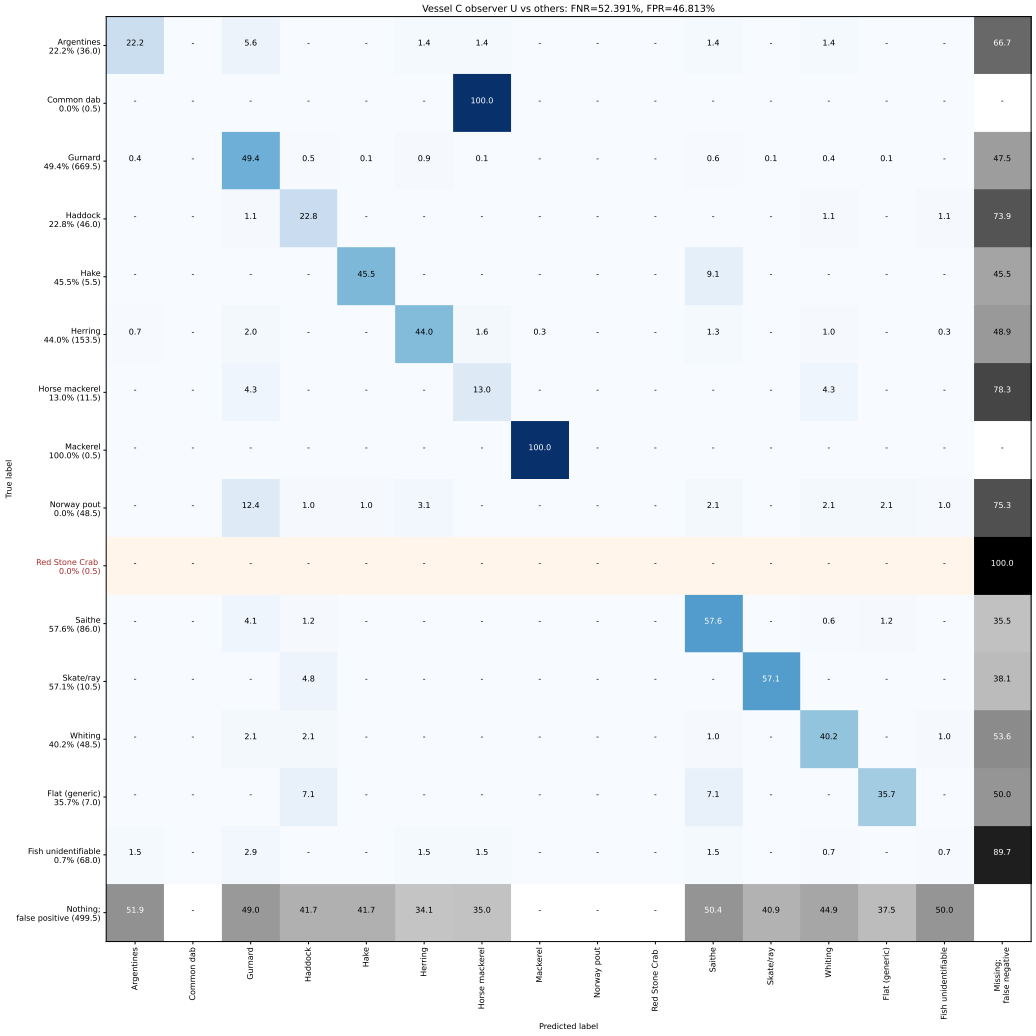


Figure B.13: Inter-observer evaluation for observer U, vessel C presented as an extended confusion matrix, contrasting the observations of observer U vs observers V and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. U count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	36.0	26.0	60.19%	29.24%	22.62%	23.74%	72.22%	30.77%	22.22%	25.81%
Common dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	669.5	685.0	80.59%	62.76%	55.87%	57.92%	97.74%	48.32%	49.44%	48.87%
Haddock	46.0	30.0	49.19%	31.55%	46.98%	33.62%	65.22%	35.00%	22.83%	27.63%
Hake	5.5	6.0	50.00%	37.50%	33.33%	28.57%	91.67%	41.67%	45.45%	43.48%
Herring	153.5	116.0	56.05%	51.70%	45.29%	41.03%	75.57%	58.19%	43.97%	50.09%
Horse mackerel	11.5	10.0	27.27%	8.33%	6.82%	5.00%	86.96%	15.00%	13.04%	13.95%
Mackerel	0.5	1.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Norway pout	48.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Red Stone Crab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	86.0	120.0	54.11%	31.81%	46.24%	34.03%	71.67%	41.25%	57.56%	48.06%
Skate/ray	10.5	11.0	44.23%	51.11%	55.13%	43.00%	95.45%	54.55%	57.14%	55.81%
Whiting	48.5	49.0	80.74%	40.34%	46.49%	42.10%	98.98%	39.80%	40.21%	40.00%
Flat (generic)	7.0	8.0	47.78%	28.33%	28.89%	22.73%	87.50%	31.25%	35.71%	33.33%
Fish unidentifiable	68.0	5.0	16.50%	8.33%	2.50%	3.57%	7.35%	10.00%	0.74%	1.37%
Any species	1192.0	1067.0	72.11%	51.89%	46.30%	47.53%	89.51%	46.81%	41.90%	44.22%
Average	1192.0	1067.0	41.11%	35.92%	32.68%	26.80%	60.02%	37.98%	32.55%	30.34%

Table B.15: Inter-observer evaluation summary for observer U, vessel C, contrasting the observations of observer U vs observers V and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Species	Other obs. count	Obs. V count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	30.0	38.0	75.34%	34.79%	43.81%	38.43%	78.95%	35.53%	45.00%	39.71%
Common dab	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	557.5	909.0	72.47%	57.27%	79.85%	65.16%	61.33%	45.71%	74.53%	56.67%
Haddock	40.5	42.0	38.02%	38.51%	46.25%	32.15%	96.43%	32.14%	33.33%	32.73%
Hake	4.5	8.0	50.00%	25.00%	50.00%	28.57%	56.25%	31.25%	55.56%	40.00%
Herring	114.5	193.0	46.93%	35.98%	74.74%	43.89%	59.33%	48.70%	82.10%	61.14%
Horse mackerel	9.0	16.0	17.78%	16.67%	31.25%	14.49%	56.25%	31.25%	55.56%	40.00%
Mackerel	0.5	1.0	50.00%	50.00%	100.00%	66.67%	50.00%	50.00%	100.00%	66.67%
Norway pout	31.0	35.0	38.81%	35.91%	17.30%	19.50%	88.57%	21.43%	24.19%	22.73%
Red Stone Crab	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Saithe	73.0	146.0	54.92%	36.63%	65.36%	41.21%	50.00%	42.12%	84.25%	56.16%
Skate/ray	9.0	14.0	38.12%	39.06%	70.00%	41.12%	64.29%	46.43%	72.22%	56.52%
Whiting	38.0	70.0	70.50%	41.38%	57.94%	46.37%	54.29%	35.71%	65.79%	46.30%
Flat (generic)	5.0	12.0	41.67%	27.78%	70.00%	33.87%	41.67%	29.17%	70.00%	41.18%
Fish unidentifiable	65.5	10.0	32.50%	10.00%	7.14%	5.88%	15.27%	10.00%	1.53%	2.65%
Any species	978.0	1496.0	64.79%	49.49%	65.73%	54.25%	65.37%	43.42%	66.41%	52.51%
Average	978.0	1496.0	41.80%	29.93%	54.90%	31.82%	51.51%	30.63%	58.77%	37.50%

Table B.16: Inter-observer evaluation summary for observer V, vessel C, contrasting the observations of observer V vs observers U and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

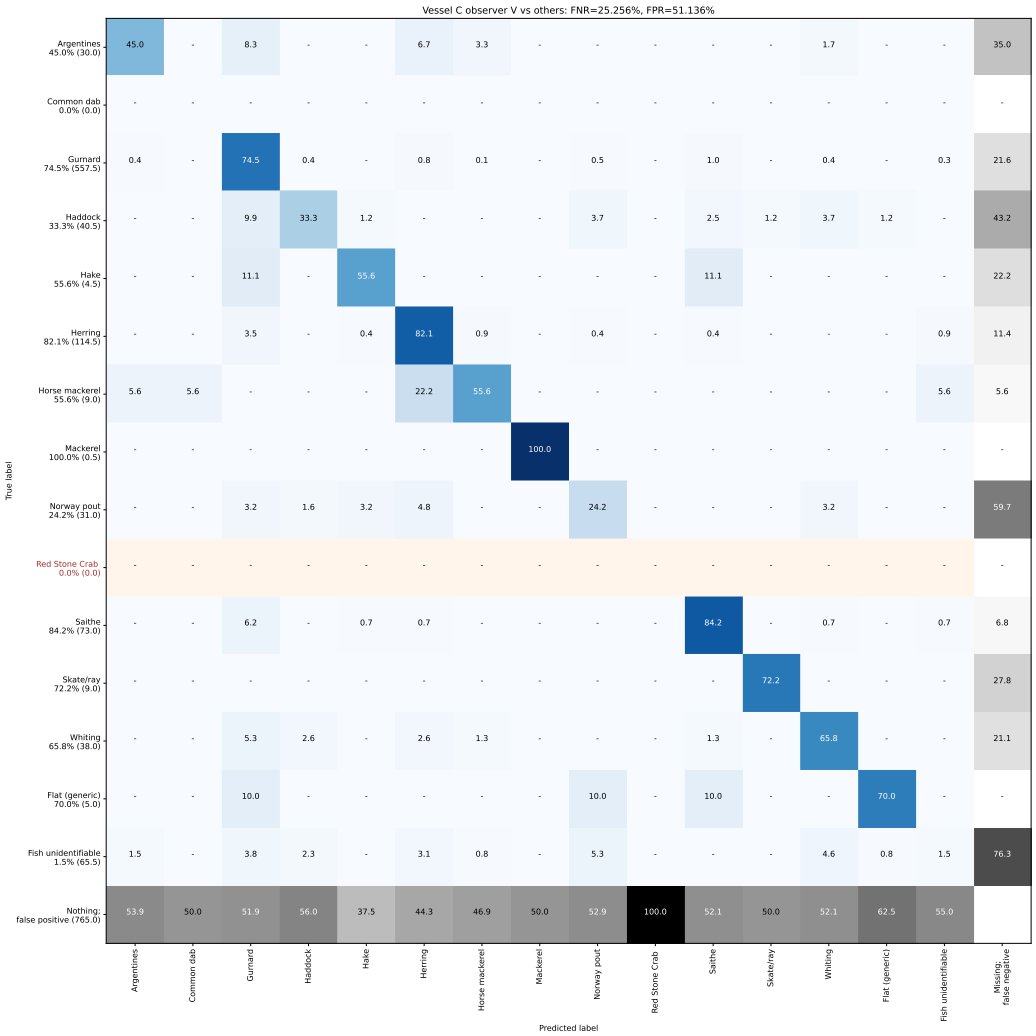


Figure B.14: Inter-observer evaluation for observer V, vessel C presented as an extended confusion matrix, contrasting the observations of observer V vs observers U and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

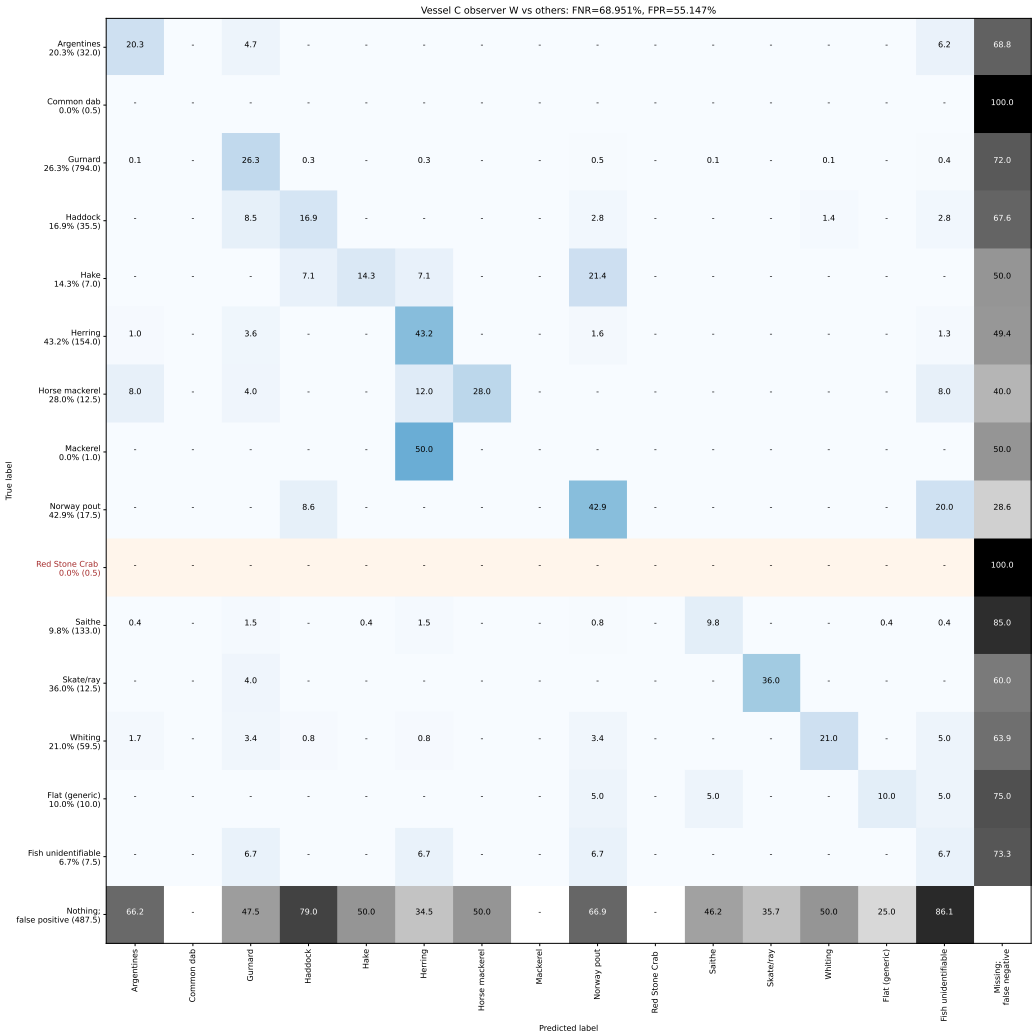


Figure B.15: Inter-observer evaluation for observer W, vessel C presented as an extended confusion matrix, contrasting the observations of observer W vs observers U and V. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. W count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Argentines	32.0	34.0	57.41%	20.14%	21.13%	18.92%	94.12%	19.12%	20.31%	19.70%
Common dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	794.0	427.0	73.34%	51.93%	39.51%	43.02%	53.78%	48.83%	26.26%	34.15%
Haddock	35.5	50.0	28.50%	53.45%	27.49%	27.76%	71.00%	12.00%	16.90%	14.04%
Hake	7.0	3.0	33.33%	25.00%	8.33%	11.11%	42.86%	33.33%	14.29%	20.00%
Herring	154.0	113.0	57.37%	52.82%	32.37%	35.22%	73.38%	58.85%	43.18%	49.81%
Horse mackerel	12.5	7.0	29.17%	50.00%	14.58%	18.42%	56.00%	50.00%	28.00%	35.90%
Mackerel	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Norway pout	17.5	62.0	16.20%	8.65%	71.83%	14.23%	28.23%	12.10%	42.86%	18.87%
Red Stone Crab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	133.0	26.0	19.71%	57.25%	12.25%	16.29%	19.55%	50.00%	9.77%	16.35%
Skate/ray	12.5	7.0	44.23%	52.50%	21.63%	23.64%	56.00%	64.29%	36.00%	46.15%
Whiting	59.5	27.0	57.39%	52.16%	29.01%	36.55%	45.38%	46.30%	21.01%	28.90%
Flat (generic)	10.0	2.0	22.22%	50.00%	11.11%	13.33%	20.00%	50.00%	10.00%	16.67%
Fish unidentifiable	7.5	126.0	29.54%	2.50%	3.57%	2.94%	5.95%	0.40%	6.67%	0.75%
Any species	1277.0	884.0	63.20%	42.23%	35.01%	35.33%	69.22%	37.44%	25.92%	30.63%
Average	1277.0	884.0	31.23%	39.70%	19.52%	17.43%	37.75%	37.10%	18.35%	20.09%

Table B.17: Inter-observer evaluation summary for observer W, vessel C, contrasting the observations of observer W vs observers U and V. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.2.4 Inter-observer variability on test set: Vessel D

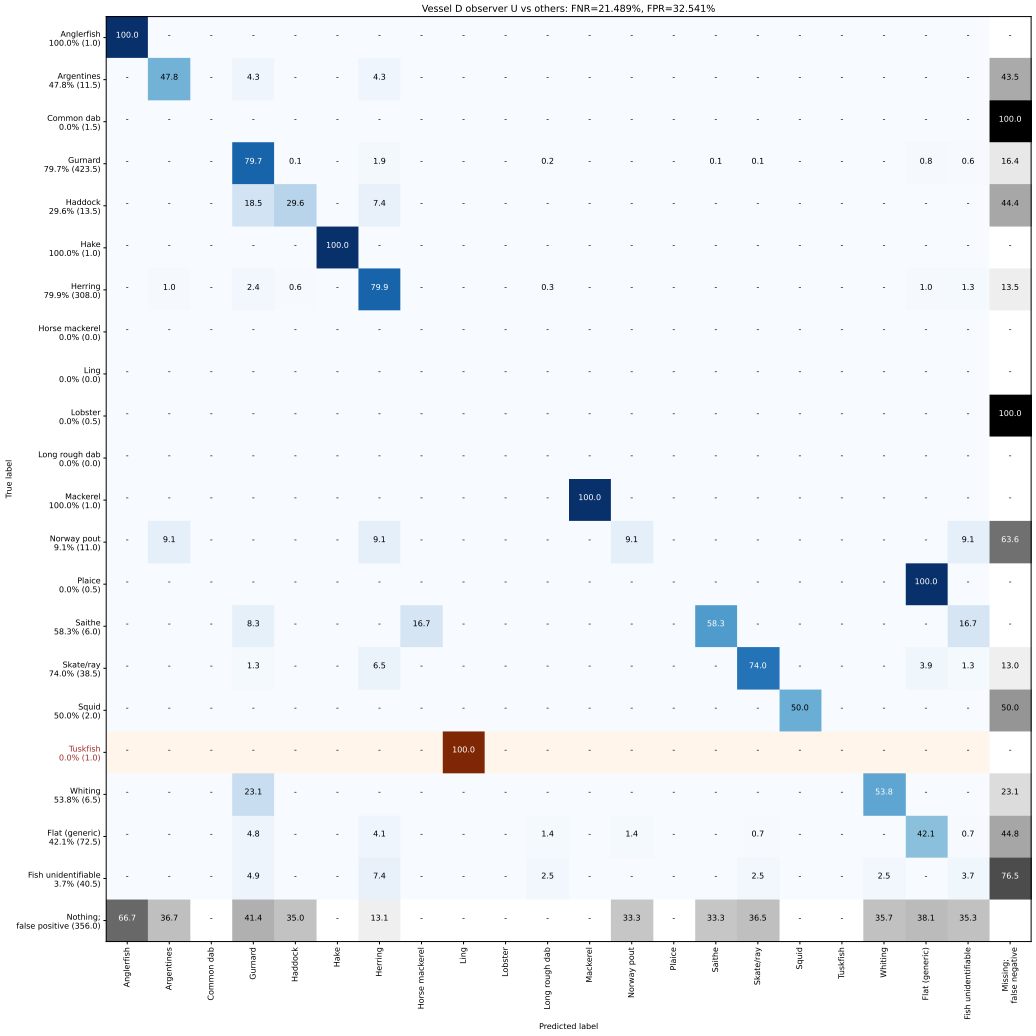


Figure B.16: Inter-observer evaluation for observer U, vessel D presented as an extended confusion matrix, contrasting the observations of observer U vs observers V and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. U count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	1.0	3.0	33.33%	33.33%	100.00%	44.44%	33.33%	33.33%	100.00%	50.00%
Argentines	11.5	15.0	38.00%	28.33%	65.00%	32.57%	76.67%	36.67%	47.83%	41.51%
Common dab	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	423.5	608.0	82.70%	57.52%	69.71%	62.29%	69.65%	55.51%	79.69%	65.44%
Haddock	13.5	10.0	31.94%	14.29%	12.50%	10.67%	74.07%	40.00%	29.63%	34.04%
Hake	1.0	1.0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Herring	308.0	305.0	83.29%	56.73%	67.93%	60.56%	99.03%	80.66%	79.87%	80.26%
Horse mackerel	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Ling	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Lobster	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	0.0	4.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Mackerel	1.0	1.0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Norway pout	11.0	3.0	5.56%	25.00%	3.70%	4.55%	27.27%	33.33%	9.09%	14.29%
Plaice	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	6.0	6.0	31.25%	58.33%	62.50%	43.33%	100.00%	58.33%	58.33%	58.33%
Skate/ray	38.5	48.0	61.96%	67.74%	69.90%	63.83%	80.21%	59.38%	74.03%	65.90%
Squid	2.0	1.0	50.00%	100.00%	50.00%	50.00%	50.00%	100.00%	50.00%	66.67%
Tuskfish	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Whiting	6.5	7.0	70.00%	55.00%	60.00%	53.33%	92.86%	50.00%	53.85%	51.85%
Flat (generic)	72.5	63.0	58.90%	50.10%	55.52%	48.02%	86.90%	48.41%	42.07%	45.02%
Fish unidentifiable	40.5	17.0	35.98%	5.33%	10.67%	7.11%	41.98%	8.82%	3.70%	5.22%
Any species	940.0	1094.0	74.62%	57.15%	64.26%	58.84%	85.92%	60.83%	70.80%	65.44%
Average	940.0	1094.0	37.28%	44.22%	45.97%	32.41%	49.14%	47.32%	46.00%	37.07%

Table B.18: Inter-observer evaluation summary for observer U, vessel D, contrasting the observations of observer U vs observers V and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

Species	Other obs. count	Obs. V count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	2.0	2.0	33.33%	50.00%	66.67%	44.44%	100.00%	50.00%	50.00%	50.00%
Argentines	13.5	14.0	48.17%	45.62%	42.38%	35.61%	96.43%	42.86%	44.44%	43.64%
Common dab	0.0	2.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	413.0	636.0	69.64%	58.33%	78.16%	64.88%	64.94%	55.19%	84.99%	66.92%
Haddock	13.0	11.0	37.50%	16.67%	11.43%	10.67%	84.62%	36.36%	30.77%	33.33%
Hake	1.0	1.0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Herring	298.5	320.0	62.99%	54.87%	80.88%	62.66%	93.28%	77.50%	83.08%	80.19%
Horse mackerel	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Ling	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Long rough dab	4.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Mackerel	1.0	1.0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Norway pout	3.5	12.0	11.81%	3.70%	16.67%	4.55%	29.17%	8.33%	28.57%	12.90%
Plaice	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Saithe	5.0	7.0	50.00%	50.00%	66.67%	45.83%	71.43%	50.00%	70.00%	58.33%
Skate/ray	34.5	54.0	68.81%	55.69%	75.06%	63.06%	63.89%	52.78%	82.61%	64.41%
Squid	1.0	3.0	50.00%	50.00%	100.00%	50.00%	33.33%	33.33%	100.00%	50.00%
Tuskfish	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Whiting	5.0	9.0	55.00%	40.00%	80.00%	51.33%	55.56%	38.89%	70.00%	50.00%
Flat (generic)	57.0	90.0	70.10%	45.33%	63.23%	51.53%	63.33%	42.78%	67.54%	52.38%
Fish unidentifiable	38.5	21.0	36.26%	12.00%	7.71%	7.79%	54.55%	9.52%	5.19%	6.72%
Any species	892.5	1186.0	74.55%	55.64%	73.09%	61.87%	75.25%	58.18%	77.31%	66.39%
Average	892.5	1186.0	37.79%	37.90%	52.29%	32.97%	48.12%	38.75%	53.95%	36.61%

Table B.19: Inter-observer evaluation summary for observer V, vessel D, contrasting the observations of observer V vs observers U and W. Please see Section 7.8.4 for an descriptions of the figures in the columns.

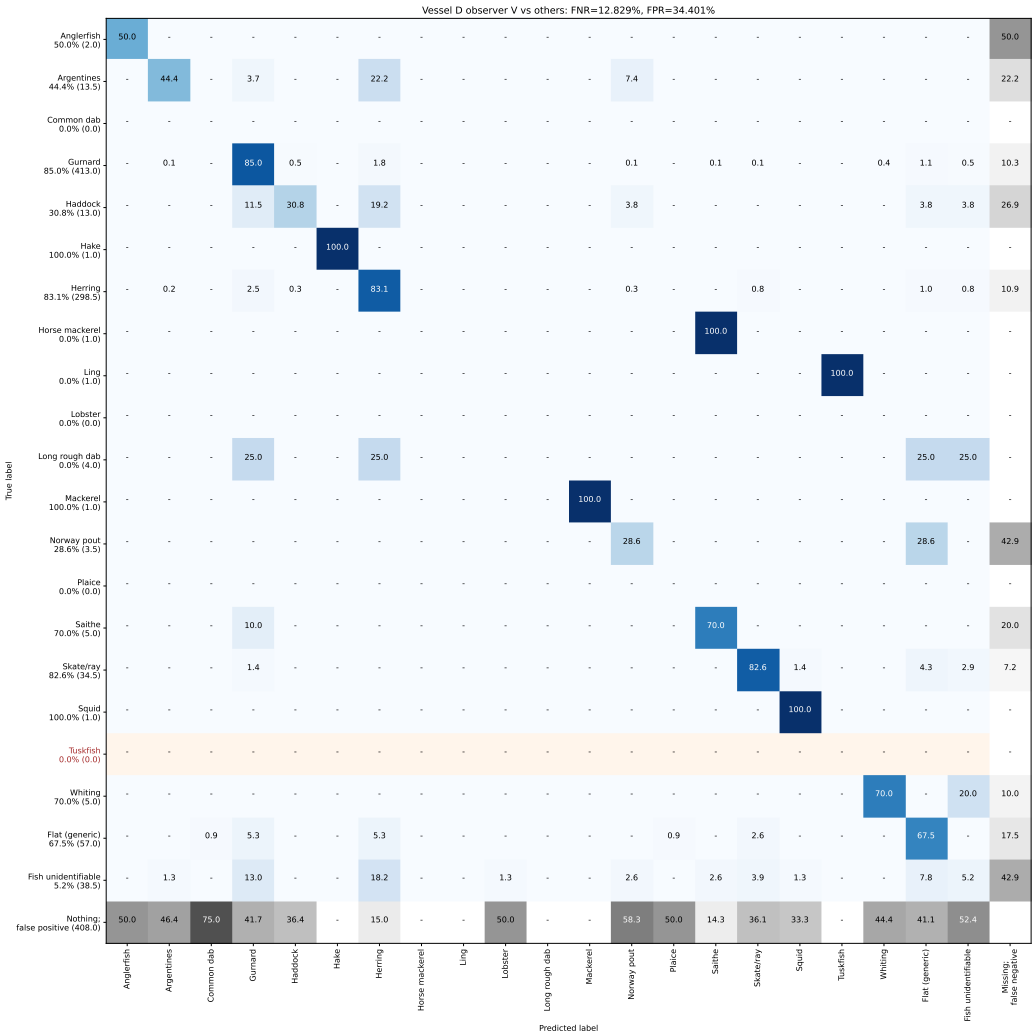


Figure B.17: Inter-observer evaluation for observer V, vessel D presented as an extended confusion matrix, contrasting the observations of observer V vs observers U and W. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

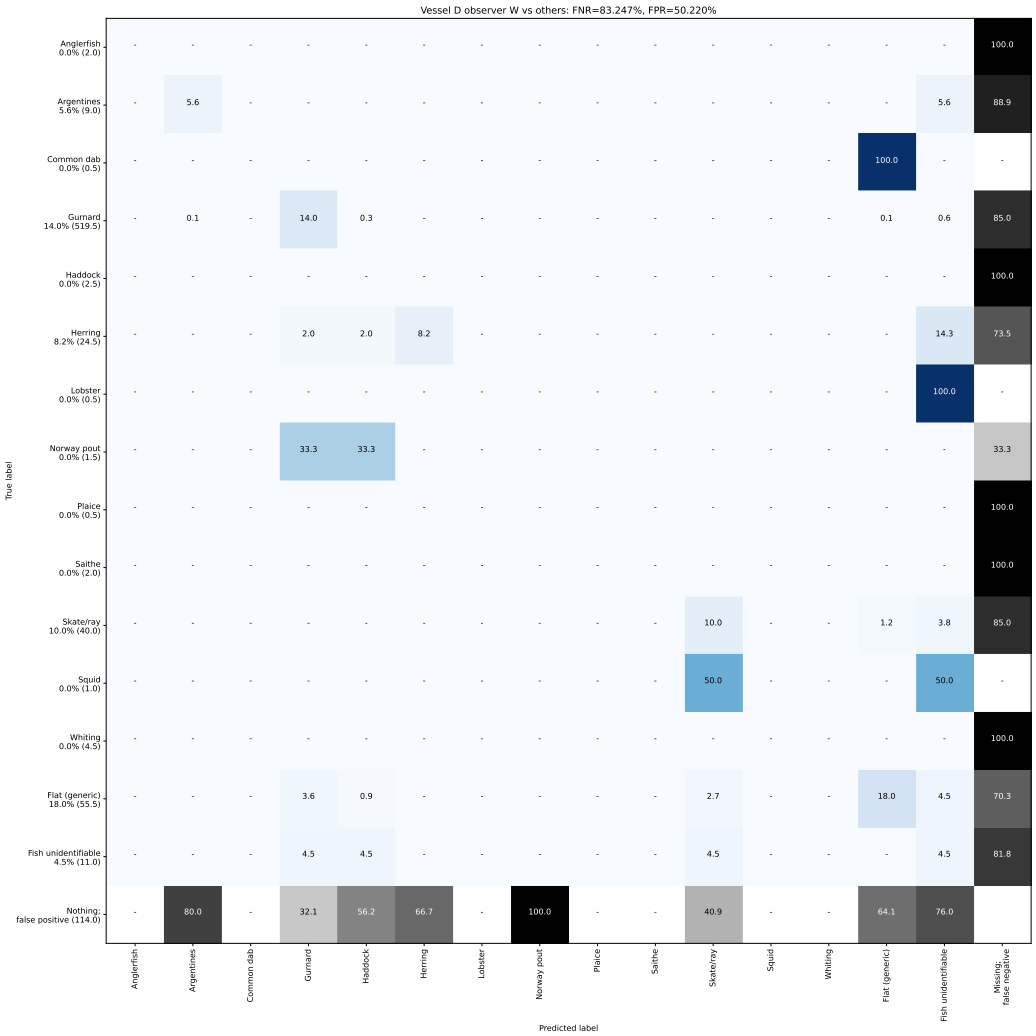


Figure B.18: Inter-observer evaluation for observer W, vessel D presented as an extended confusion matrix, contrasting the observations of observer W vs observers U and V. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. W count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Argentines	9.0	5.0	23.33%	10.00%	4.76%	3.92%	55.56%	10.00%	5.56%	7.14%
Common dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	519.5	112.0	30.02%	50.81%	11.92%	18.89%	21.56%	64.73%	13.96%	22.96%
Haddock	2.5	8.0	7.14%	0.00%	0.00%	0.00%	31.25%	0.00%	0.00%	0.00%
Hake	0.0	0.0	–	–	–	–	–	–	–	–
Herring	24.5	6.0	17.09%	33.33%	5.88%	9.29%	24.49%	33.33%	8.16%	13.11%
Horse mackerel	0.0	0.0	–	–	–	–	–	–	–	–
Ling	0.0	0.0	–	–	–	–	–	–	–	–
Lobster	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	0.0	0.0	–	–	–	–	–	–	–	–
Mackerel	0.0	0.0	–	–	–	–	–	–	–	–
Norway pout	1.5	1.0	50.00%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Plaice	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Saithe	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	40.0	11.0	33.48%	28.57%	4.85%	7.73%	27.50%	36.36%	10.00%	15.69%
Squid	1.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Tuskfish	0.0	0.0	–	–	–	–	–	–	–	–
Whiting	4.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Flat (generic)	55.5	32.0	37.81%	42.86%	19.71%	21.48%	57.66%	31.25%	18.02%	22.86%
Fish unidentifiable	11.0	52.0	24.22%	0.50%	5.56%	0.63%	21.15%	0.96%	4.55%	1.59%
Any species	674.5	227.0	38.73%	46.13%	12.91%	17.80%	33.65%	39.43%	13.27%	19.86%
Average	674.5	227.0	14.87%	20.76%	3.51%	4.13%	20.39%	22.08%	4.02%	5.56%

Table B.20: Inter-observer evaluation summary for observer W, vessel D, contrasting the observations of observer W vs observers U and V. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.2.5 Inter-observer variability on test set: Vessel E

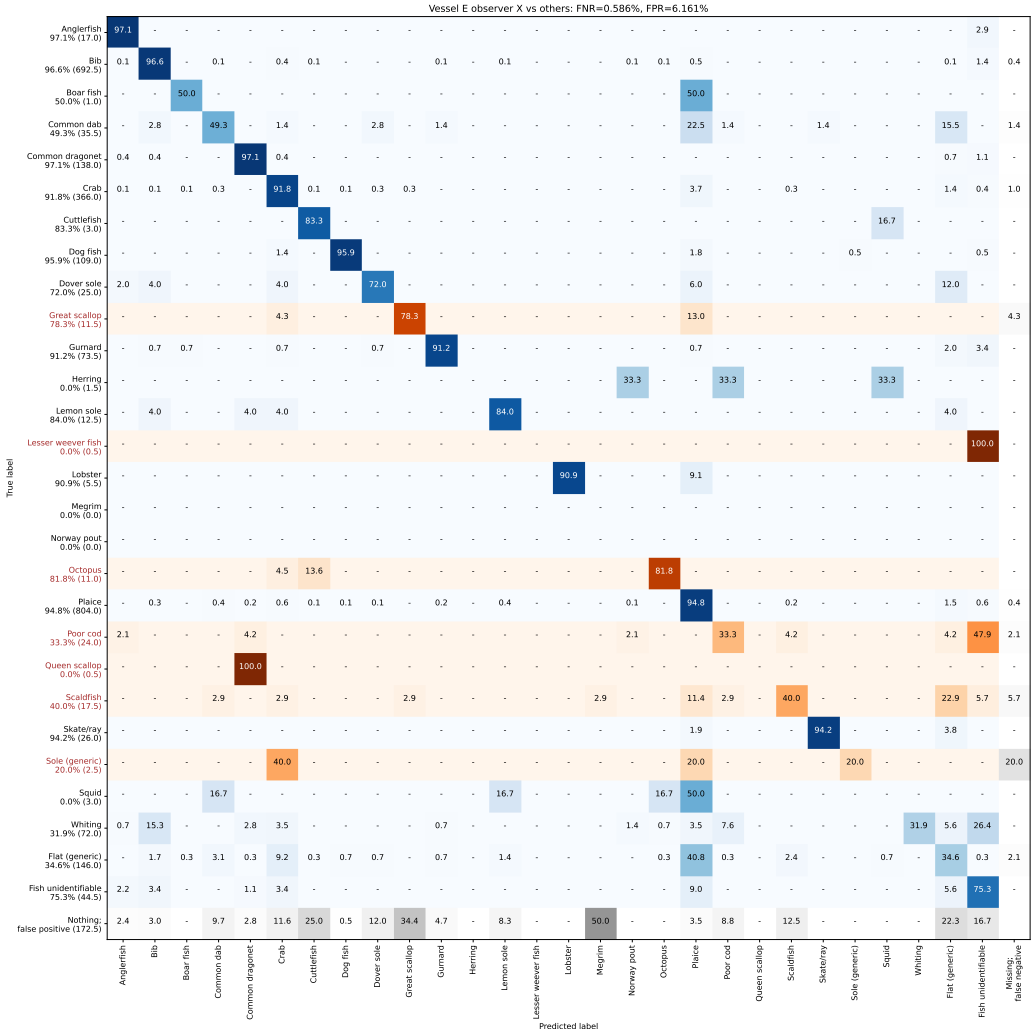


Figure B.19: Inter-observer evaluation for observer X, vessel E presented as an extended confusion matrix, contrasting the observations of observer X vs observers Y and Z. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. X count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	17.0	21.0	80.83%	79.79%	98.75%	86.87%	80.95%	78.57%	97.06%	86.84%
Bib	692.5	712.0	93.17%	95.78%	94.22%	94.57%	97.26%	93.96%	96.61%	95.27%
Boar fish	1.0	2.0	50.00%	25.00%	50.00%	25.00%	50.00%	25.00%	50.00%	33.33%
Common dab	35.5	31.0	54.20%	57.59%	47.35%	45.22%	87.32%	56.45%	49.30%	52.63%
Common dragonet	138.0	145.0	89.37%	90.90%	94.14%	91.39%	95.17%	92.41%	97.10%	94.70%
Crab	366.0	417.0	85.20%	80.80%	88.07%	83.48%	87.77%	80.58%	91.80%	85.82%
Cuttlefish	3.0	8.0	25.00%	30.00%	75.00%	33.33%	37.50%	31.25%	83.33%	45.45%
Dog fish	109.0	107.0	94.48%	89.29%	87.99%	88.62%	98.17%	97.66%	95.87%	96.76%
Dover sole	25.0	25.0	57.75%	57.01%	85.74%	60.20%	100.00%	72.00%	72.00%	72.00%
Great scallop	11.5	16.0	54.86%	51.67%	52.00%	41.97%	71.88%	56.25%	78.26%	65.45%
Gurnard	73.5	75.0	76.02%	82.57%	79.30%	74.86%	98.00%	89.33%	91.16%	90.24%
Herring	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lemon sole	12.5	18.0	71.43%	64.58%	86.19%	67.66%	69.44%	58.33%	84.00%	68.85%
Lesser weever fish	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	5.5	5.0	83.33%	100.00%	83.33%	83.33%	90.91%	100.00%	90.91%	95.24%
Megrim	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Norway pout	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Octopus	11.0	11.0	60.71%	82.50%	69.64%	62.84%	100.00%	81.82%	81.82%	81.82%
Plaice	804.0	895.0	83.63%	81.48%	96.20%	86.96%	89.83%	85.14%	94.78%	89.70%
Poor cod	24.0	17.0	41.13%	37.50%	16.86%	19.83%	70.83%	47.06%	33.33%	39.02%
Queen scallop	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Scaldfish	17.5	16.0	43.52%	45.83%	31.55%	29.55%	91.43%	43.75%	40.00%	41.79%
Skate/ray	26.0	25.0	98.21%	96.88%	95.09%	95.91%	96.15%	98.00%	94.23%	96.08%
Sole (generic)	2.5	1.0	25.00%	50.00%	12.50%	12.50%	40.00%	50.00%	20.00%	28.57%
Squid	3.0	2.0	0.00%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Whiting	72.0	23.0	21.07%	100.00%	21.07%	32.40%	31.94%	100.00%	31.94%	48.42%
Flat (generic)	146.0	119.0	53.02%	41.50%	46.61%	35.70%	81.51%	42.44%	34.59%	38.11%
Fish unidentifiable	44.5	105.0	43.43%	26.68%	44.14%	25.99%	42.38%	31.90%	75.28%	44.82%
Any species	2643.0	2800.0	92.13%	76.66%	83.19%	79.59%	94.39%	82.43%	87.33%	84.81%
Average	2643.0	2800.0	49.48%	58.69%	55.99%	45.65%	63.40%	60.48%	60.90%	53.25%

Table B.21: Inter-observer evaluation summary for observer X, vessel E, contrasting the observations of observer X vs observers Y and Z. Please see Section 7.8.4 for an descriptions of the figures in the columns.

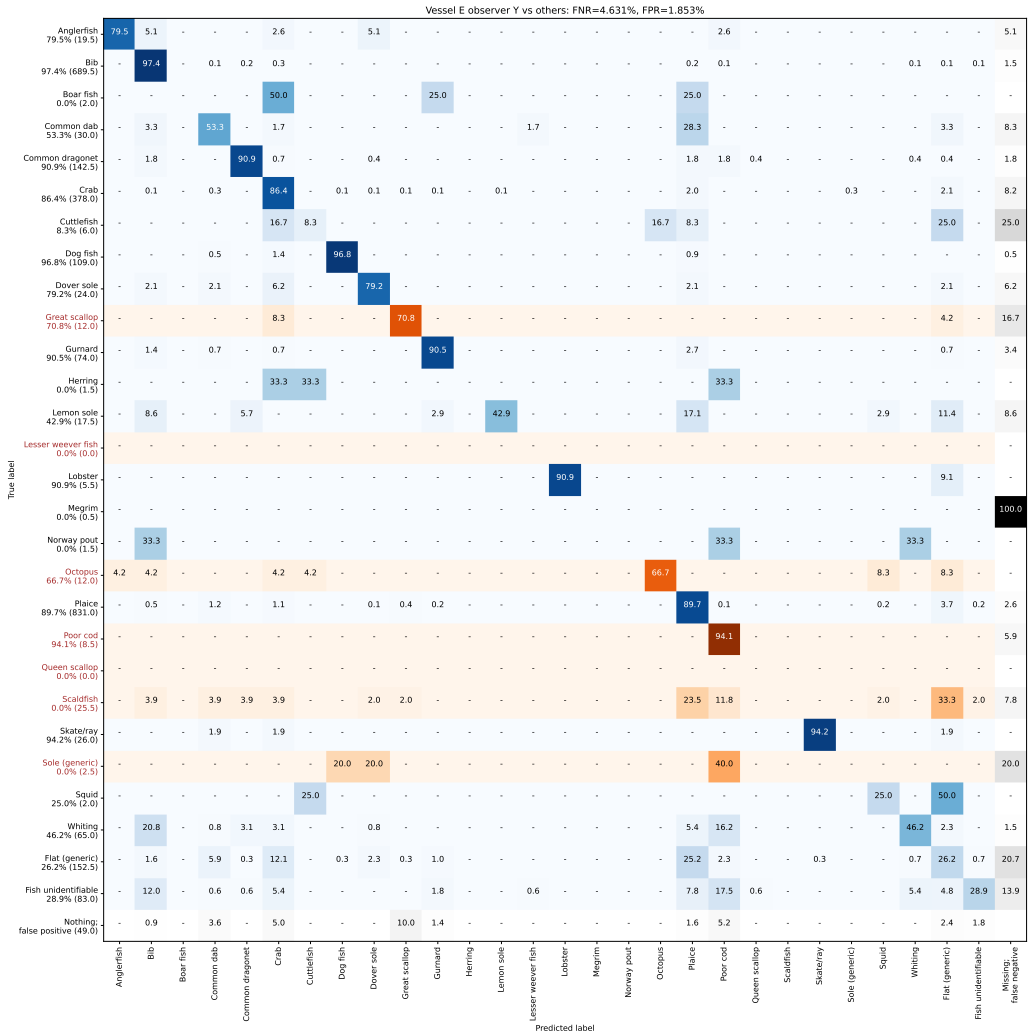


Figure B.20: Inter-observer evaluation for observer Y, vessel E presented as an extended confusion matrix, contrasting the observations of observer Y vs observers X and Z. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. Y count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	19.5	16.0	79.86%	93.75%	86.11%	87.60%	82.05%	96.88%	79.49%	87.32%
Bib	689.5	718.0	92.00%	94.80%	95.18%	94.45%	96.03%	93.52%	97.39%	95.42%
Boar fish	2.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	30.0	42.0	48.40%	40.52%	64.24%	43.84%	71.43%	38.10%	53.33%	44.44%
Common dragonet	142.5	136.0	89.97%	93.43%	91.10%	91.46%	95.44%	95.22%	90.88%	93.00%
Crab	378.0	393.0	83.45%	86.53%	80.70%	81.27%	96.18%	83.08%	86.38%	84.70%
Cuttlefish	6.0	2.0	8.33%	25.00%	20.00%	11.11%	33.33%	25.00%	8.33%	12.50%
Dog fish	109.0	107.0	90.79%	97.99%	88.94%	89.37%	98.17%	98.60%	96.79%	97.69%
Dover sole	24.0	27.0	52.20%	71.12%	66.06%	56.00%	88.89%	70.37%	79.17%	74.51%
Great scallop	12.0	15.0	43.33%	53.75%	46.08%	39.49%	80.00%	56.67%	70.83%	62.96%
Gurnard	74.0	74.0	76.39%	81.93%	84.23%	79.06%	100.00%	90.54%	90.54%	90.54%
Herring	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lemon sole	17.5	8.0	47.25%	94.44%	43.68%	53.20%	45.71%	93.75%	42.86%	58.82%
Lesser weever fish	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Lobster	5.5	5.0	83.33%	100.00%	83.33%	83.33%	90.91%	100.00%	90.91%	95.24%
Megrim	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Norway pout	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Octopus	12.0	9.0	54.76%	83.33%	63.89%	60.95%	75.00%	88.89%	66.67%	76.19%
Plaice	831.0	841.0	87.33%	88.04%	86.21%	86.39%	98.81%	88.64%	89.71%	89.17%
Poor cod	8.5	48.0	10.28%	8.43%	75.00%	14.32%	17.71%	16.67%	94.12%	28.32%
Queen scallop	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Scaldfish	25.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	26.0	25.0	91.61%	99.11%	92.50%	94.18%	96.15%	98.00%	94.23%	96.08%
Sole (generic)	2.5	1.0	0.00%	0.00%	0.00%	0.00%	40.00%	0.00%	0.00%	0.00%
Squid	2.0	4.0	20.00%	16.67%	25.00%	13.33%	50.00%	12.50%	25.00%	16.67%
Whiting	65.0	37.0	46.08%	78.50%	60.46%	55.62%	56.92%	81.08%	46.15%	58.82%
Flat (generic)	152.5	106.0	58.30%	41.51%	27.18%	29.27%	69.51%	37.74%	26.23%	30.95%
Fish unidentifiable	83.0	28.0	38.45%	60.42%	17.32%	24.82%	33.73%	85.71%	28.92%	43.24%
Any species	2721.0	2644.0	94.07%	80.88%	77.63%	79.15%	97.17%	85.19%	82.78%	83.97%
Average	2721.0	2644.0	42.93%	61.27%	49.89%	42.47%	54.14%	63.08%	52.23%	47.74%

Table B.22: Inter-observer evaluation summary for observer Y, vessel E, contrasting the observations of observer Y vs observers X and Z. Please see Section 7.8.4 for an descriptions of the figures in the columns.

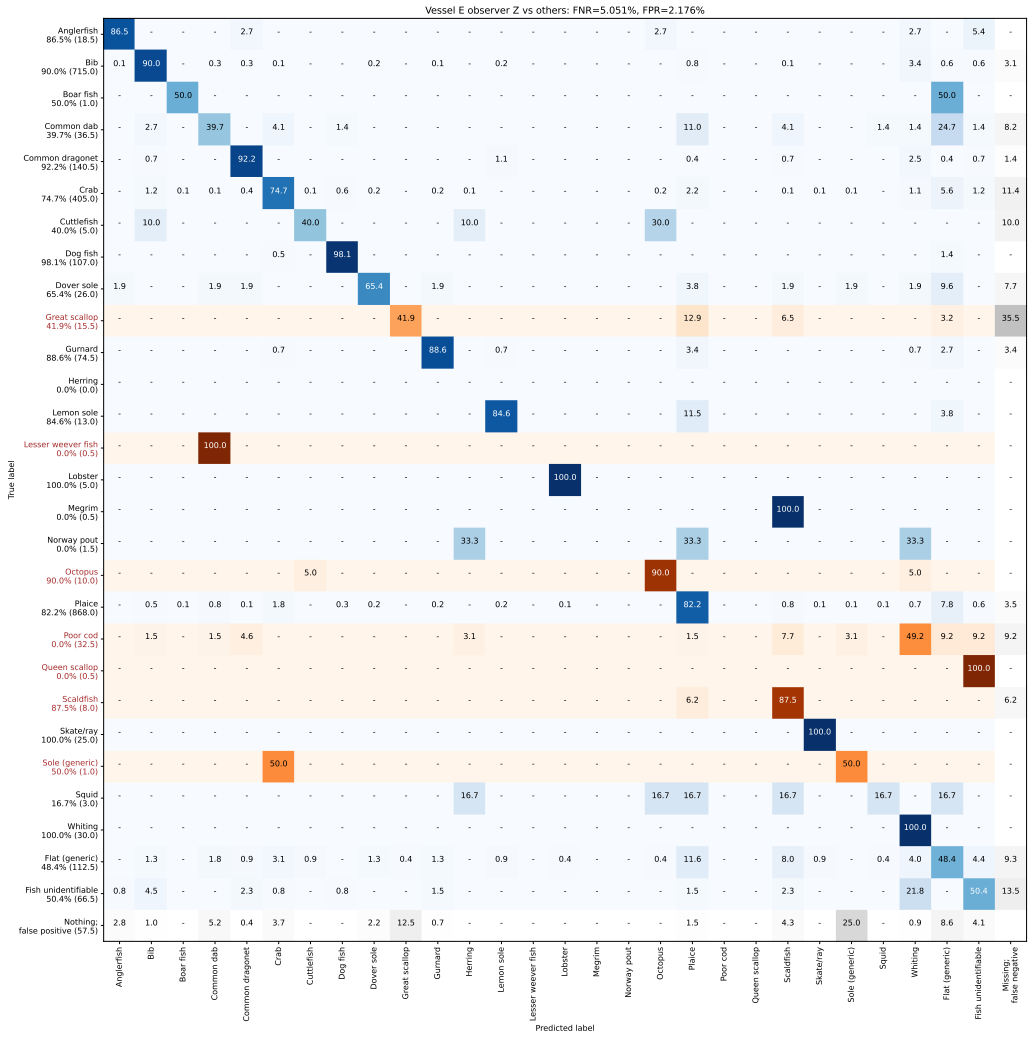


Figure B.21: Inter-observer evaluation for observer Z, vessel E presented as an extended confusion matrix, contrasting the observations of observer Z vs observers X and Y. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. Z count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	18.5	18.0	77.78%	94.05%	80.69%	80.86%	97.30%	88.89%	86.49%	87.67%
Bib	715.0	667.0	87.51%	93.14%	92.29%	91.49%	93.29%	96.48%	90.00%	93.13%
Boar fish	1.0	2.0	12.50%	25.00%	50.00%	20.00%	50.00%	25.00%	50.00%	33.33%
Common dab	36.5	29.0	54.51%	49.77%	41.94%	37.77%	79.45%	50.00%	39.73%	44.27%
Common dragonet	140.5	140.0	89.84%	95.48%	92.14%	93.10%	99.64%	92.50%	92.17%	92.34%
Crab	405.0	339.0	79.75%	84.00%	79.15%	78.65%	83.70%	89.23%	74.69%	81.32%
Cuttlefish	5.0	4.0	16.67%	50.00%	33.33%	22.22%	80.00%	50.00%	40.00%	44.44%
Dog fish	107.0	111.0	93.72%	86.18%	88.19%	87.13%	96.40%	94.59%	98.13%	96.33%
Dover sole	26.0	23.0	47.80%	80.95%	48.72%	48.13%	88.46%	73.91%	65.38%	69.39%
Great scallop	15.5	8.0	35.35%	72.50%	41.52%	39.38%	51.61%	81.25%	41.94%	55.32%
Gurnard	74.5	73.0	83.13%	81.42%	80.03%	78.29%	97.99%	90.41%	88.59%	89.49%
Herring	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Lemon sole	13.0	17.0	67.11%	62.95%	83.33%	70.22%	76.47%	64.71%	84.62%	73.33%
Lesser weever fish	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	5.0	6.0	83.33%	83.33%	100.00%	83.33%	83.33%	83.33%	100.00%	90.91%
Megrim	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Norway pout	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Octopus	10.0	13.0	58.81%	74.17%	77.78%	64.01%	76.92%	69.23%	90.00%	78.26%
Plaice	868.0	767.0	82.68%	92.78%	78.19%	83.39%	88.36%	93.02%	82.20%	87.28%
Poor cod	32.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Queen scallop	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Scaldfish	8.0	35.0	17.96%	15.77%	91.67%	24.30%	22.86%	20.00%	87.50%	32.56%
Skate/ray	25.0	27.0	92.86%	92.86%	100.00%	95.37%	92.59%	92.59%	100.00%	96.15%
Sole (generic)	1.0	4.0	6.25%	8.33%	50.00%	10.00%	25.00%	12.50%	50.00%	20.00%
Squid	3.0	2.0	20.00%	25.00%	20.00%	13.33%	66.67%	25.00%	16.67%	20.00%
Whiting	30.0	107.0	35.20%	35.20%	100.00%	48.27%	28.04%	28.04%	100.00%	43.80%
Flat (generic)	112.5	186.0	45.51%	38.61%	41.63%	32.43%	60.48%	29.30%	48.44%	36.52%
Fish unidentifiable	66.5	61.0	43.78%	38.90%	44.39%	29.62%	91.73%	54.92%	50.38%	52.55%
Any species	2722.0	2642.0	94.00%	78.03%	74.86%	76.30%	97.06%	82.99%	80.55%	81.75%
Average	2722.0	2642.0	44.00%	60.02%	56.11%	43.98%	58.22%	61.08%	58.40%	50.66%

Table B.23: Inter-observer evaluation summary for observer Z, vessel E, contrasting the observations of observer Z vs observers X and Y. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.2.6 Inter-observer variability on test set: Vessel F

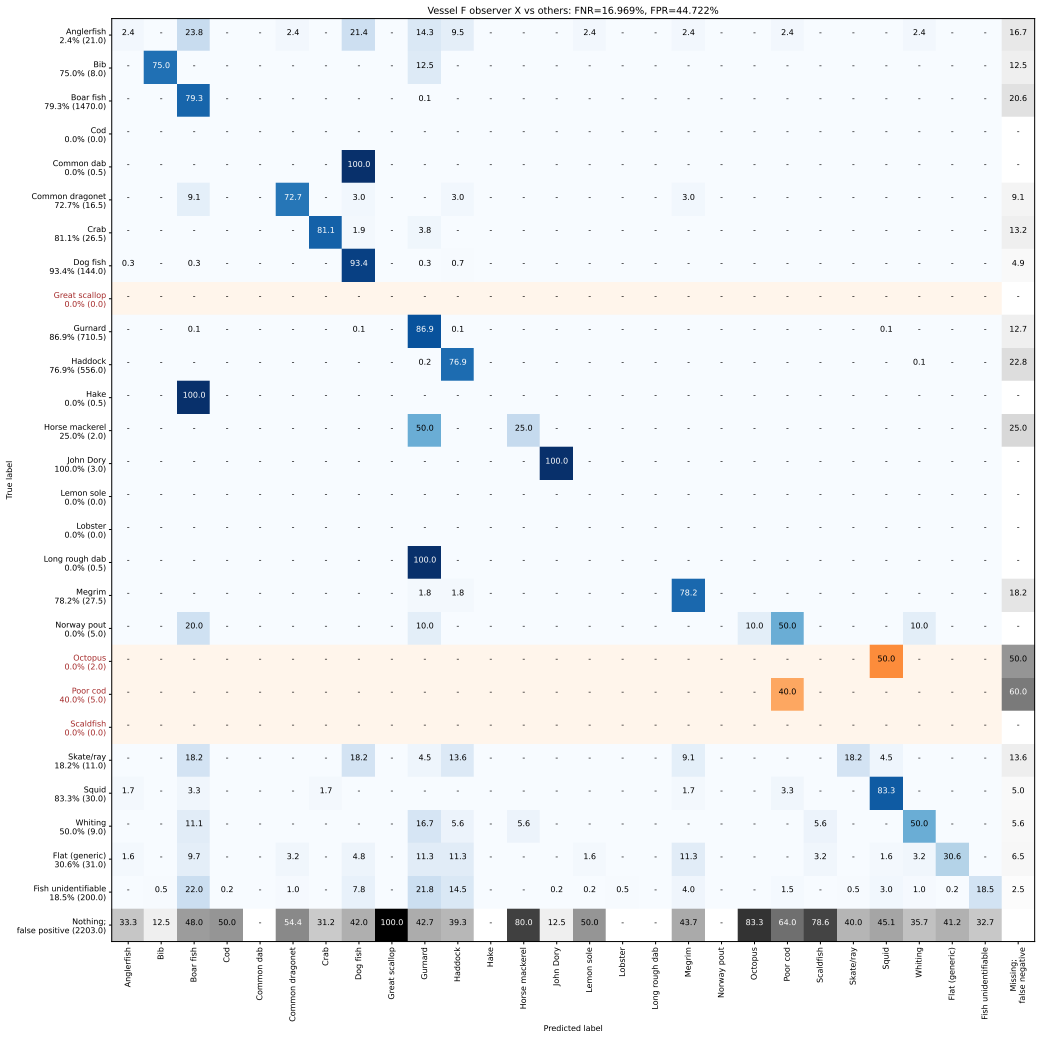


Figure B.22: Inter-observer evaluation for observer X, vessel F presented as an extended confusion matrix, contrasting the observations of observer X vs observers Y and Z. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. X count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	21.0	3.0	10.21%	16.67%	1.04%	1.79%	14.29%	16.67%	2.38%	4.17%
Bib	8.0	8.0	83.33%	66.67%	64.44%	64.85%	100.00%	75.00%	75.00%	75.00%
Boar fish	1470.0	2359.0	58.44%	54.55%	93.14%	68.38%	62.31%	49.43%	79.32%	60.90%
Cod	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Common dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	16.5	34.0	61.89%	58.22%	77.74%	61.41%	48.53%	35.29%	72.73%	47.52%
Crab	26.5	32.0	80.61%	78.98%	87.92%	81.54%	82.81%	67.19%	81.13%	73.50%
Dog fish	144.0	276.0	63.24%	56.85%	91.82%	68.65%	52.17%	48.73%	93.40%	64.05%
Great scallop	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Gurnard	710.5	1181.0	60.69%	57.56%	94.76%	71.06%	60.16%	52.29%	86.91%	65.29%
Haddock	556.0	768.0	70.88%	63.33%	90.34%	74.18%	72.40%	55.66%	76.89%	64.58%
Hake	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Horse mackerel	2.0	5.0	33.33%	8.33%	16.67%	9.52%	40.00%	10.00%	25.00%	14.29%
John Dory	3.0	4.0	75.00%	75.00%	100.00%	84.13%	75.00%	75.00%	100.00%	85.71%
Lemon sole	0.0	3.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Lobster	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Long rough dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Megrim	27.5	63.0	39.38%	33.08%	90.43%	45.39%	43.65%	34.13%	78.18%	47.51%
Norway pout	5.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Octopus	2.0	3.0	66.67%	0.00%	0.00%	0.00%	66.67%	0.00%	0.00%	0.00%
Poor cod	5.0	25.0	20.83%	16.07%	80.00%	24.13%	20.00%	8.00%	40.00%	13.33%
Scaldfish	0.0	7.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Skate/ray	11.0	5.0	30.95%	44.44%	17.86%	17.62%	45.45%	40.00%	18.18%	25.00%
Squid	30.0	61.0	65.91%	58.02%	85.53%	65.62%	49.18%	40.98%	83.33%	54.95%
Whiting	9.0	14.0	43.39%	28.70%	66.27%	33.24%	64.29%	32.14%	50.00%	39.13%
Flat (generic)	31.0	17.0	47.11%	69.44%	29.93%	37.38%	54.84%	55.88%	30.65%	39.58%
Fish unidentifiable	200.0	55.0	26.12%	75.26%	19.38%	29.80%	27.50%	67.27%	18.50%	29.02%
Any species	3279.5	4926.0	68.53%	57.57%	83.59%	67.90%	66.58%	50.56%	75.94%	60.70%
Average	3279.5	4926.0	34.74%	37.44%	50.33%	31.06%	36.27%	33.20%	45.98%	29.76%

Table B.24: Inter-observer evaluation summary for observer X, vessel F, contrasting the observations of observer X vs observers Y and Z. Please see Section 7.8.4 for an descriptions of the figures in the columns.

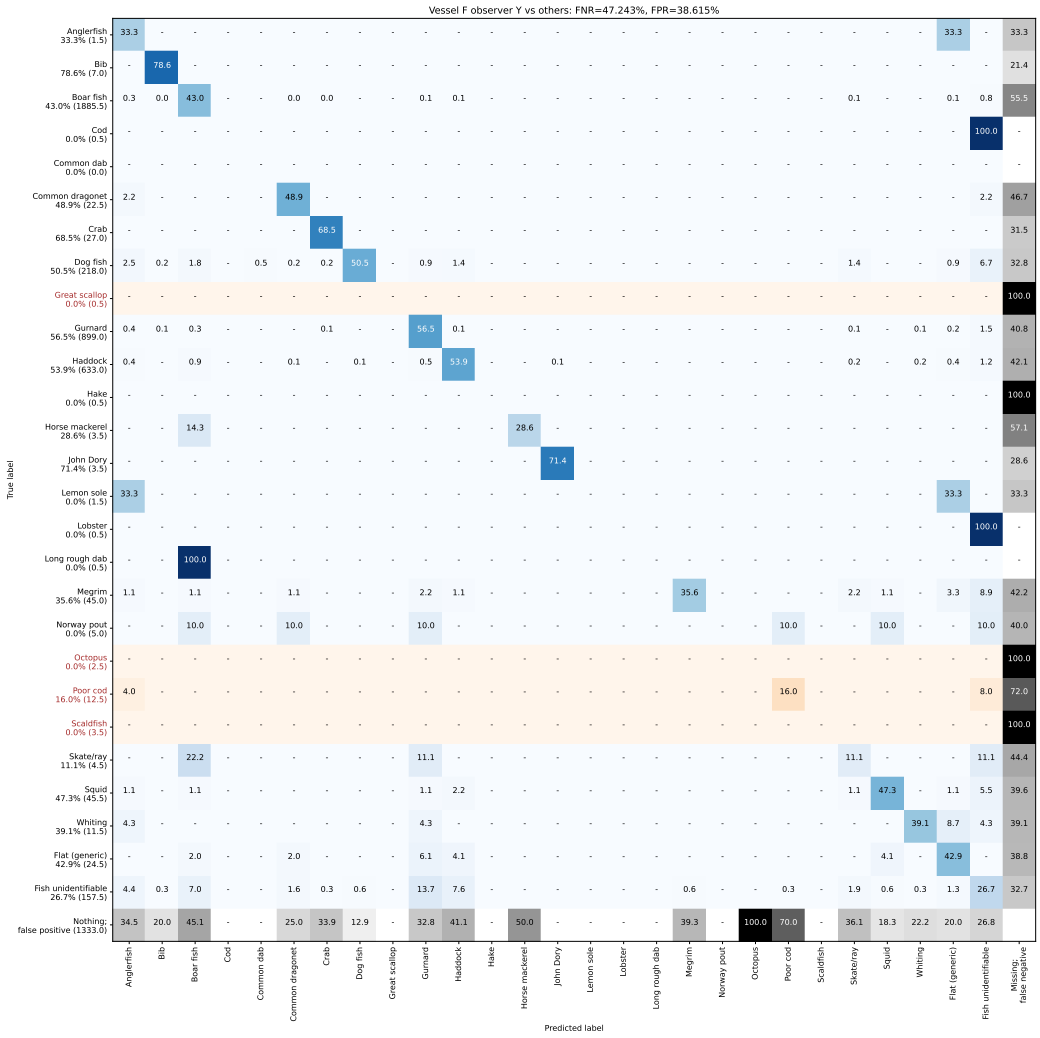


Figure B.23: Inter-observer evaluation for observer Y, vessel F presented as an extended confusion matrix, contrasting the observations of observer Y vs observers X and Z. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. Y count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	1.5	42.0	2.55%	0.52%	33.33%	1.00%	3.57%	1.19%	33.33%	2.30%
Bib	7.0	10.0	66.67%	45.83%	58.33%	50.00%	70.00%	55.00%	78.57%	64.71%
Boar fish	1885.5	1528.0	85.98%	72.90%	66.74%	69.37%	81.04%	53.08%	43.01%	47.52%
Cod	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Common dragonet	22.5	22.0	64.38%	54.17%	75.00%	59.05%	97.78%	50.00%	48.89%	49.44%
Crab	27.0	31.0	77.99%	67.66%	88.19%	75.34%	87.10%	59.68%	68.52%	63.79%
Dog fish	218.0	128.0	67.42%	80.39%	56.53%	64.93%	58.72%	85.94%	50.46%	63.58%
Great scallop	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	899.0	804.0	82.41%	74.98%	71.62%	72.48%	89.43%	63.18%	56.51%	59.66%
Haddock	633.0	614.0	88.53%	77.37%	72.38%	74.35%	97.00%	55.62%	53.95%	54.77%
Hake	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Horse mackerel	3.5	2.0	50.00%	50.00%	16.67%	22.22%	57.14%	50.00%	28.57%	36.36%
John Dory	3.5	3.0	58.33%	87.50%	66.67%	61.90%	85.71%	83.33%	71.43%	76.92%
Lemon sole	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Megrim	45.0	28.0	57.00%	75.95%	37.83%	46.38%	62.22%	57.14%	35.56%	43.84%
Norway pout	5.0	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Octopus	2.5	2.0	80.00%	0.00%	0.00%	0.00%	80.00%	0.00%	0.00%	0.00%
Poor cod	12.5	10.0	53.57%	40.00%	32.14%	28.57%	80.00%	20.00%	16.00%	17.78%
Scaldfish	3.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	4.5	18.0	30.48%	3.57%	6.67%	3.17%	25.00%	2.78%	11.11%	4.44%
Squid	45.5	30.0	80.12%	74.38%	62.50%	66.83%	65.93%	71.67%	47.25%	56.95%
Whiting	11.5	9.0	45.24%	54.17%	38.62%	35.09%	78.26%	50.00%	39.13%	43.90%
Flat (generic)	24.5	30.0	58.80%	37.76%	56.29%	41.54%	81.67%	35.00%	42.86%	38.53%
Fish unidentifiable	157.5	140.0	48.14%	32.88%	41.08%	30.08%	88.89%	30.00%	26.67%	28.24%
Any species	4016.5	3452.0	88.56%	70.63%	67.14%	68.58%	85.95%	55.23%	47.47%	51.05%
Average	4016.5	3452.0	40.65%	48.95%	33.87%	29.71%	47.76%	43.35%	28.92%	27.88%

Table B.25: Inter-observer evaluation summary for observer Y, vessel F, contrasting the observations of observer Y vs observers X and Z. Please see Section 7.8.4 for an descriptions of the figures in the columns.

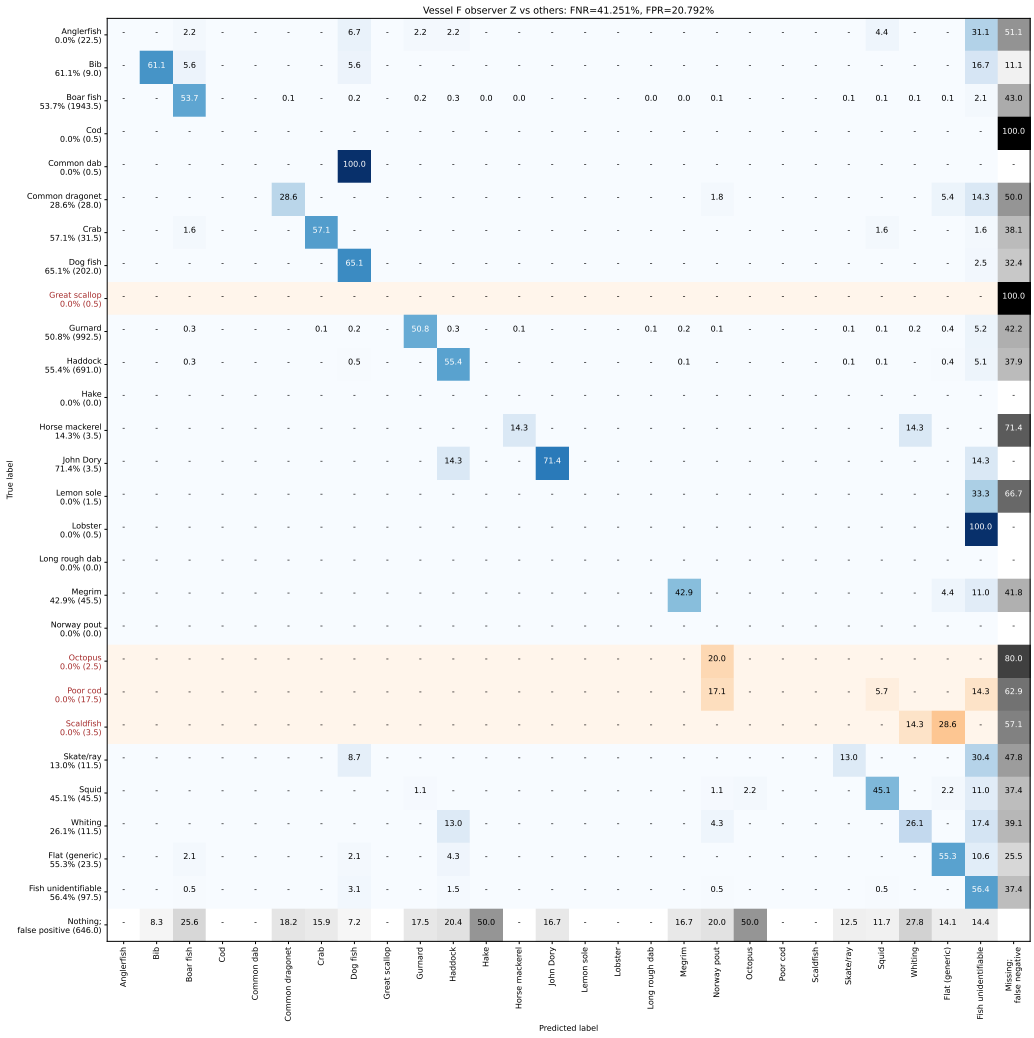


Figure B.24: Inter-observer evaluation for observer Z, vessel F presented as an extended confusion matrix, contrasting the observations of observer Z vs observers X and Y. Orange rows indicate species that were not present in the training set, hence we cannot expect correct predictions from the model. The last row and column are in greyscale and represent respectively the false positives and false negatives that arise from differing assessment by observers.

Species	Other obs. count	Obs. Z count	Video count accuracy	Video precision	Video recall	Video F1	Count accuracy	Vessel precision	Vessel recall	Vessel F1
Anglerfish	22.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Bib	9.0	6.0	65.08%	83.33%	53.97%	61.48%	66.67%	91.67%	61.11%	73.33%
Boar fish	1943.5	1412.0	60.30%	91.40%	54.00%	67.12%	72.65%	73.87%	53.67%	62.17%
Cod	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dab	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Common dragonet	28.0	11.0	44.92%	85.00%	39.56%	42.93%	39.29%	72.73%	28.57%	41.03%
Crab	31.5	22.0	66.63%	91.84%	61.37%	68.82%	69.84%	81.82%	57.14%	67.29%
Dog fish	202.0	160.0	78.39%	82.64%	66.80%	73.13%	79.21%	82.19%	65.10%	72.65%
Great scallop	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Gurnard	992.5	617.0	59.72%	93.69%	54.66%	68.31%	62.17%	81.77%	50.83%	62.69%
Haddock	691.0	498.0	71.64%	88.83%	62.85%	72.96%	72.07%	76.91%	55.43%	64.42%
Hake	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Horse mackerel	3.5	2.0	25.00%	25.00%	11.11%	9.52%	57.14%	25.00%	14.29%	18.18%
John Dory	3.5	3.0	66.67%	83.33%	83.33%	77.78%	85.71%	83.33%	71.43%	76.92%
Lemon sole	1.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Lobster	0.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Long rough dab	0.0	1.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Megrim	45.5	27.0	50.87%	83.73%	39.31%	48.45%	59.34%	72.22%	42.86%	53.79%
Norway pout	0.0	10.0	0.00%	0.00%	–	0.00%	0.00%	0.00%	–	0.00%
Octopus	2.5	2.0	80.00%	0.00%	0.00%	0.00%	80.00%	0.00%	0.00%	0.00%
Poor cod	17.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Scaldfish	3.5	0.0	0.00%	–	0.00%	0.00%	0.00%	–	0.00%	0.00%
Skate/ray	11.5	4.0	32.86%	41.67%	13.57%	15.08%	34.78%	37.50%	13.04%	19.35%
Squid	45.5	30.0	72.34%	77.34%	54.96%	60.67%	65.93%	68.33%	45.05%	54.30%
Whiting	11.5	9.0	32.22%	60.00%	16.67%	22.22%	78.26%	33.33%	26.09%	29.27%
Flat (generic)	23.5	32.0	53.14%	50.00%	59.52%	49.24%	73.44%	40.62%	55.32%	46.85%
Fish unidentifiable	97.5	260.0	49.81%	29.39%	59.89%	33.98%	37.50%	21.15%	56.41%	30.77%
Any species	4189.0	3107.0	70.82%	81.12%	56.77%	66.40%	74.17%	71.10%	52.73%	60.55%
Average	4189.0	3107.0	33.69%	56.17%	30.48%	28.58%	38.30%	49.60%	29.01%	28.63%

Table B.26: Inter-observer evaluation summary for observer Z, vessel F, contrasting the observations of observer Z vs observers X and Y. Please see Section 7.8.4 for an descriptions of the figures in the columns.

B.2.7 Inter-observer variability on test set: video summaries

MSS videos

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel A	Video 0	Good	Easy	1024.0	1094.0	93.60%	53.29%	56.93%	55.05%
	Video 1	Good	Hard	653.5	745.0	87.72%	41.88%	47.74%	44.62%
	Video 2	Good	Moderate	246.0	234.0	95.12%	43.80%	41.67%	42.71%
	Video 3	Poor	Moderate	263.5	247.0	93.74%	26.92%	25.24%	26.05%
	Video 4	Fair	Moderate	344.5	383.0	89.95%	41.64%	46.30%	43.85%
	Average			506.3	540.6	92.03%	41.51%	43.58%	42.46%
Vessel B	Video 0	Good	Hard	182.5	247.0	73.89%	39.27%	53.15%	45.17%
	Video 1	Poor	Moderate	66.5	125.0	53.20%	45.20%	84.96%	59.01%
	Video 2	Good	Moderate	456.0	448.0	98.25%	37.39%	36.73%	37.06%
	Video 3	Good	Easy	304.5	615.0	49.51%	45.77%	92.45%	61.23%
	Video 4	Fair	Moderate	556.5	770.0	72.27%	46.30%	64.06%	53.75%
	Average			313.2	441.0	69.42%	42.79%	66.27%	51.24%
Vessel C	Video 0	Poor	Moderate	175.0	136.0	77.71%	50.00%	38.86%	43.73%
	Video 1	Good	Hard	235.0	126.0	53.62%	65.08%	34.89%	45.43%
	Video 2	Good	Easy	4.0	5.0	80.00%	60.00%	75.00%	66.67%
	Video 3	Good	Moderate	301.0	207.0	68.77%	39.86%	27.41%	32.48%
	Video 4	Fair	Moderate	477.0	593.0	80.44%	44.52%	55.35%	49.35%
	Average			238.4	213.4	72.11%	51.89%	46.30%	47.53%
Vessel D	Video 0	Good	Hard	193.0	205.0	94.15%	74.15%	78.76%	76.38%
	Video 1	Fair	Moderate	126.0	164.0	76.83%	42.99%	55.95%	48.62%
	Video 2	Poor	Moderate	231.0	407.0	56.76%	52.33%	92.21%	66.77%
	Video 3	Good	Easy	274.0	259.0	94.53%	80.69%	76.28%	78.42%
	Video 4	Good	Moderate	116.0	59.0	50.86%	35.59%	18.10%	24.00%
	Average			188.0	218.8	74.62%	57.15%	64.26%	58.84%

Table B.27: Per-video inter-observer summary for observer U vs. observers V and W, vessels A-D

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel A	Video 0	Good	Easy	975.0	1192.0	81.80%	53.23%	65.08%	58.56%
	Video 1	Good	Hard	699.0	654.0	93.56%	43.20%	40.41%	41.76%
	Video 2	Good	Moderate	178.0	370.0	48.11%	40.27%	83.71%	54.38%
	Video 3	Poor	Moderate	238.5	297.0	80.30%	27.61%	34.38%	30.63%
	Video 4	Fair	Moderate	294.5	483.0	60.97%	39.23%	64.35%	48.75%
	Average			477.0	599.2	72.95%	40.71%	57.59%	46.81%
Vessel B	Video 0	Good	Hard	232.0	148.0	63.79%	43.24%	27.59%	33.68%
	Video 1	Poor	Moderate	84.5	89.0	94.94%	44.38%	46.75%	45.53%
	Video 2	Good	Moderate	418.5	523.0	80.02%	41.87%	52.33%	46.52%
	Video 3	Good	Easy	353.0	518.0	68.15%	47.10%	69.12%	56.03%
	Video 4	Fair	Moderate	636.0	611.0	96.07%	48.04%	46.15%	47.07%
	Average			344.8	377.8	80.59%	44.93%	48.39%	45.77%
Vessel C	Video 0	Poor	Moderate	135.5	215.0	63.02%	48.84%	77.49%	59.91%
	Video 1	Good	Hard	158.5	279.0	56.81%	42.29%	74.45%	53.94%
	Video 2	Good	Easy	5.0	4.0	80.00%	75.00%	60.00%	66.67%
	Video 3	Good	Moderate	294.5	220.0	74.70%	37.50%	28.01%	32.07%
	Video 4	Fair	Moderate	384.5	778.0	49.42%	43.83%	88.69%	58.67%
	Average			195.6	299.2	64.79%	49.49%	65.73%	54.25%
Vessel D	Video 0	Good	Hard	205.0	193.0	94.15%	78.76%	74.15%	76.38%
	Video 1	Fair	Moderate	102.5	211.0	48.58%	41.23%	84.88%	55.50%
	Video 2	Poor	Moderate	239.0	391.0	61.13%	53.32%	87.24%	66.19%
	Video 3	Good	Easy	259.0	274.0	94.53%	76.28%	80.69%	78.42%
	Video 4	Good	Moderate	87.0	117.0	74.36%	28.63%	38.51%	32.84%
	Average			178.5	237.2	74.55%	55.64%	73.09%	61.87%

Table B.28: Per-video inter-observer summary for observer V vs. observers U and W, vessels A-D

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel A	Video 0	Good	Easy	1143.0	856.0	74.89%	51.11%	38.28%	43.77%
	Video 1	Good	Hard	699.5	653.0	93.35%	41.73%	38.96%	40.30%
	Video 2	Good	Moderate	302.0	122.0	40.40%	38.11%	15.40%	21.93%
	Video 3	Poor	Moderate	272.0	230.0	84.56%	33.70%	28.49%	30.88%
	Video 4	Fair	Moderate	433.0	206.0	47.58%	34.47%	16.40%	22.22%
	Average			569.9	413.4	68.15%	39.82%	27.50%	31.82%
Vessel B	Video 0	Good	Hard	197.5	217.0	91.01%	41.01%	45.06%	42.94%
	Video 1	Poor	Moderate	107.0	44.0	41.12%	45.45%	18.69%	26.49%
	Video 2	Good	Moderate	485.5	389.0	80.12%	39.72%	31.82%	35.33%
	Video 3	Good	Easy	566.5	91.0	16.06%	41.21%	6.62%	11.41%
	Video 4	Fair	Moderate	690.5	502.0	72.70%	46.81%	34.03%	39.41%
	Average			409.4	248.6	60.20%	42.84%	27.25%	31.12%
Vessel C	Video 0	Poor	Moderate	175.5	135.0	76.92%	55.56%	42.74%	48.31%
	Video 1	Good	Hard	202.5	191.0	94.32%	45.55%	42.96%	44.22%
	Video 2	Good	Easy	nan	nan	–	–	–	–
	Video 3	Good	Moderate	213.5	382.0	55.89%	24.08%	43.09%	30.90%
	Video 4	Fair	Moderate	685.5	176.0	25.67%	43.75%	11.23%	17.88%
	Average			319.25	221.0	63.20%	42.23%	35.01%	35.33%
Vessel D	Video 0	Good	Hard	nan	nan	–	–	–	–
	Video 1	Fair	Moderate	187.5	41.0	21.87%	45.12%	9.87%	16.19%
	Video 2	Poor	Moderate	399.0	71.0	17.79%	82.39%	14.66%	24.89%
	Video 3	Good	Easy	nan	nan	–	–	–	–
	Video 4	Good	Moderate	88.0	115.0	76.52%	10.87%	14.20%	12.32%
	Average			224.83	75.67	38.73%	46.13%	12.91%	17.80%

Table B.29: Per-video inter-observer summary for observer W vs. observers U and V, vessels A-D. Three videos; one from vessel C and two from vessel D were not assesed by observer W, hence the lack of values in the corresponding rows.

Cefas videos

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel E	Video 0	Poor	Moderate	98.0	103.0	95.15%	85.92%	90.31%	88.06%
	Video 1	Poor	Moderate	394.0	415.0	94.94%	88.07%	92.77%	90.36%
	Video 2	Moderate	Easy	141.0	178.0	79.21%	72.19%	91.13%	80.56%
	Video 3	Poor	Moderate	62.0	68.0	91.18%	70.59%	77.42%	73.85%
	Video 4	Poor	Moderate	86.0	86.0	100.00%	63.37%	63.37%	63.37%
	Video 5	Moderate	Moderate	479.5	493.0	97.26%	87.73%	90.20%	88.95%
	Video 6	Moderate	Moderate-easy	266.0	281.0	94.66%	80.25%	84.77%	82.45%
	Video 7	Good	Moderate	459.0	484.0	94.83%	84.50%	89.11%	86.74%
	Video 8	Good	Moderate	28.5	43.0	66.28%	51.16%	77.19%	61.54%
	Video 9	Good	Moderate	166.0	167.0	99.40%	84.13%	84.64%	84.38%
	Video 10	Poor	Moderate	159.0	165.0	96.36%	80.00%	83.02%	81.48%
	Video 11	Good	Moderate-easy	12.0	13.0	92.31%	65.38%	70.83%	68.00%
	Video 12	Good	Moderate-easy	292.0	304.0	96.05%	83.22%	86.64%	84.90%
	Average			203.307692	215.384615	92.13%	76.66%	83.19%	79.59%
Vessel F	Video 0	Fair	Moderate-easy	1178.5	1890.0	62.35%	32.25%	51.72%	39.73%
	Video 1	Fair	Hard	635.5	815.0	77.98%	72.45%	92.92%	81.42%
	Video 2	Fair	Moderate	282.5	389.0	72.62%	67.61%	93.10%	78.33%
	Video 3	Good	Moderate	250.0	414.0	60.39%	52.66%	87.20%	65.66%
	Video 4	Good	Moderate-easy	201.0	273.0	73.63%	67.22%	91.29%	77.43%
	Video 5	Good	Hard	139.0	208.0	66.83%	57.69%	86.33%	69.16%
	Video 6	Good	Moderate	336.0	499.0	67.33%	59.32%	88.10%	70.90%
	Video 7	Good	Easy	75.0	90.0	83.33%	65.56%	78.67%	71.52%
	Video 8	Moderate	Hard	182.0	348.0	52.30%	43.39%	82.97%	56.98%
	Average			364.389	547.33	68.53%	57.57%	83.59%	67.90%

Table B.30: Per-video inter-observer summary for observer X vs. observers Y and Z, vessels E-F

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel E	Video 0	Poor	Moderate	101.0	97.0	96.04%	90.21%	86.63%	88.38%
	Video 1	Poor	Moderate	406.0	391.0	96.31%	92.20%	88.79%	90.46%
	Video 2	Moderate	Easy	158.5	143.0	90.22%	82.52%	74.45%	78.28%
	Video 3	Poor	Moderate	66.0	60.0	90.91%	71.67%	65.15%	68.25%
	Video 4	Poor	Moderate	88.0	82.0	93.18%	67.68%	63.07%	65.29%
	Video 5	Moderate	Moderate	489.5	473.0	96.63%	89.53%	86.52%	88.00%
	Video 6	Moderate	Moderate-easy	274.5	264.0	96.17%	83.52%	80.33%	81.89%
	Video 7	Good	Moderate	462.5	477.0	96.96%	84.17%	86.81%	85.47%
	Video 8	Good	Moderate	35.5	29.0	81.69%	67.24%	54.93%	60.47%
	Video 9	Good	Moderate	168.0	163.0	97.02%	80.98%	78.57%	79.76%
	Video 10	Poor	Moderate	163.5	156.0	95.41%	83.97%	80.12%	82.00%
	Video 11	Good	Moderate-easy	12.0	13.0	92.31%	73.08%	79.17%	76.00%
	Video 12	Good	Moderate-easy	296.0	296.0	100.00%	84.63%	84.63%	84.63%
	Average			209.307692	203.384615	94.07%	80.88%	77.63%	79.15%
Vessel F	Video 0	Fair	Moderate-easy	1565.0	1117.0	71.37%	4.12%	2.94%	3.43%
	Video 1	Fair	Hard	711.5	663.0	93.18%	86.73%	80.82%	83.67%
	Video 2	Fair	Moderate	305.5	343.0	89.07%	77.99%	87.56%	82.50%
	Video 3	Good	Moderate	300.0	314.0	95.54%	68.15%	71.33%	69.71%
	Video 4	Good	Moderate-easy	223.0	229.0	97.38%	79.26%	81.39%	80.31%
	Video 5	Good	Hard	165.5	155.0	93.66%	76.13%	71.30%	73.63%
	Video 6	Good	Moderate	401.5	368.0	91.66%	79.48%	72.85%	76.02%
	Video 7	Good	Easy	81.5	77.0	94.48%	85.06%	80.37%	82.65%
	Video 8	Moderate	Hard	263.0	186.0	70.72%	78.76%	55.70%	65.26%
	Average			446.278	383.56	88.56%	70.63%	67.14%	68.58%

Table B.31: Per-video inter-observer summary for observer Y vs. observers X and Z, vessels E-F

Vessel	Video	Quality	Difficulty	True count	Pred. count	Count accuracy	Precision	Recall	F1
Vessel E	Video 0	Poor	Moderate	100.0	99.0	99.00%	87.88%	87.00%	87.44%
	Video 1	Poor	Moderate	403.0	397.0	98.51%	88.41%	87.10%	87.75%
	Video 2	Moderate	Easy	160.5	139.0	86.60%	85.25%	73.83%	79.13%
	Video 3	Poor	Moderate	64.0	64.0	100.00%	68.75%	68.75%	68.75%
	Video 4	Poor	Moderate	84.0	90.0	93.33%	61.11%	65.48%	63.22%
	Video 5	Moderate	Moderate	483.0	486.0	99.38%	84.98%	85.51%	85.24%
	Video 6	Moderate	Moderate-easy	272.5	268.0	98.35%	78.73%	77.43%	78.08%
	Video 7	Good	Moderate	480.5	441.0	91.78%	87.64%	80.44%	83.88%
	Video 8	Good	Moderate	36.0	28.0	77.78%	48.21%	37.50%	42.19%
	Video 9	Good	Moderate	165.0	169.0	97.63%	79.59%	81.52%	80.54%
	Video 10	Poor	Moderate	160.5	162.0	99.07%	77.16%	77.88%	77.52%
	Video 11	Good	Moderate-easy	13.0	11.0	84.62%	81.82%	69.23%	75.00%
	Video 12	Good	Moderate-easy	300.0	288.0	96.00%	84.90%	81.50%	83.16%
	Average			209.384615	203.230769	94.00%	78.03%	74.86%	76.30%
Vessel F	Video 0	Fair	Moderate-easy	1503.5	1240.0	82.47%	48.10%	39.67%	43.48%
	Video 1	Fair	Hard	739.0	608.0	82.27%	88.57%	72.87%	79.96%
	Video 2	Fair	Moderate	366.0	222.0	60.66%	96.17%	58.33%	72.62%
	Video 3	Good	Moderate	364.0	186.0	51.10%	89.78%	45.88%	60.73%
	Video 4	Good	Moderate-easy	251.0	173.0	68.92%	91.33%	62.95%	74.53%
	Video 5	Good	Hard	181.5	123.0	67.77%	85.37%	57.85%	68.97%
	Video 6	Good	Moderate	433.5	304.0	70.13%	82.40%	57.79%	67.93%
	Video 7	Good	Easy	83.5	73.0	87.43%	80.14%	70.06%	74.76%
	Video 8	Moderate	Hard	267.0	178.0	66.67%	68.26%	45.51%	54.61%
	Average			465.44	345.22	70.82%	81.12%	56.77%	66.40%

Table B.32: Per-video inter-observer summary for observer Z vs. observers X and Y, vessels E-F