Data Efficient Deep Learning Algorithms for Video Processing Systems

Artjoms Gorpincenko

A thesis presented for the degree of
Doctor of Philosophy

School of Computer Science
University of East Anglia
United Kingdom
December 2022

Data Efficient Deep Learning Algorithms for Video Processing Systems

Artjoms Gorpincenko

2022

Abstract

Over the last decade, deep neural networks have grown in popularity and became the standard approach for the majority of computer vision tasks. Their ability to deliver state of the art results without involving carefully designed hand-crafted features is unmatched, however, that strong performance comes at the cost of requiring large amounts of labelled training data. For many modern applications, collecting and annotating new data can often be time-consuming and expensive, if not impossible altogether.

In this thesis, we focus on scenarios where the cost of acquiring image and video samples and ground truths is the main barrier. We explore different ways to artificially expand the available training sets and propose methods which aim to boost the performance and robustness of neural networks in such situations.

We first visit the field of generative adversarial models and utilise one to mimic static sonar images. Combined with a few other techniques, such as a video event classifier, weighted loss, and confidence threshold, it results in a more accurate and generalised system that can be used to solve a real-world problem. Then, we explore the idea of expanding the training set directly in feature space. We apply virtual adversarial training to video descriptors

and observe an improvement in performance in the unsupervised video domain adaptation setting. Finally, we pay close attention to the temporal domain and utilise it to introduce a large number of video-specific transformations, along with a magnitude augmentation framework. The obtained results show that time domain consideration while designing video transformations is beneficial for networks that aim to solve video action recognition.

Data Efficient Deep Learning Algorithms for Video Processing Systems

Artjoms Gorpincenko

Declaration

I hereby declare that this thesis has been composed by myself and that it has not been previously submitted to any other institution for a degree or qualification. Parts of the thesis have been published in authoritative conferences or journals. I can confirm that all the papers listed below, as the research results of my PhD, were written by me - Artjoms Gorpincenko.

Gorpincenko, A., French, G., Knight, P., Challiss, M., and Mackiewicz, M. (2021). Improving automated sonar video analysis to notify about jellyfish blooms. *IEEE Sensors Journal*, 21(4):4981-4988. (Chapter 3)

Gorpincenko, A., French, G., and Mackiewicz, M. (2021). Virtual adversarial training in feature space to improve unsupervised video domain adaptation. *Electronic Imaging*, 2021(10):258-1-258-6. (Chapter 4)

Gorpincenko, A. and Mackiewicz, M. (2021). Svw-ucf dataset for video domain adaptation. In *Proceedings of the International Conference on Image Processing and Vision Engineering - IMPROVE*, pages 107-111. INSTICC, SciTePress. (Chapter 4)

Gorpincenko, A. and Mackiewicz, M. (2022). Extending temporal data augmentation for video recognition. To be published in *Proceedings of the 37th International Conference on Image and Vision Computing New Zealand - IVCNZ*. Springer. (Chapter 5)

Acknowledgements

I would like to thank my supervisor, Dr. Michal Mackiewicz for his help, assistance, and encouragement throughout this project. I would like to say thank you to Dr. Ben Milner for his support, advice, and teaching opportunities during my time at the University of East Anglia. I would like to thank Geoff French for his input and fruitful discussions about the research.

I would like to say thank you to Chloe Game, Jake McVey, and Brandon Hobley. It seems like it was only yesterday that we started this journey together in an empty lab. You have always been there for me, through all the highs and lows, for which I am forever grateful.

I would like to thank Peter Knight, Mike Challiss, Brian Robinson, and Julie Bremner from Cefas and Cefas Technology Limited for their involvement in the JellyMonitor project discussed in Chapter 3.

This project was supported by the Natural Environment Research Council (NERC), Engineering and Physical Sciences Research Council (EPSRC), and Cefas through the NEXUSS Centre for Doctoral Training Industrial Case (iCASE) grant (grant #NE/RO12156/1).

Contents

List of Figures

List of Tables

List of Acronyms

CE	Conditional Entropy
CNN	Convolutional Neural Network
DA	Domain Adaptation
DNN	Deep Neural Network
EMA	Exponential Moving Average
FC	Fully-Connected Layer
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HMDB	Human Motion Database
HOG	Histogram of Oriented Gradients
LOOCV	Leave-One-Out Cross-Validation
MLP	Multilayer Perceptron
SIFT	Scale-Invariant Feature Transform
SOTA	State of The Art
SSL	Semi-Supervised Learning
SVW	Sports Videos in The Wild
TA^{3}N	Temporal Attentive Adversarial Adaptation Network
UDA	Unsupervised Domain Adaptation
UEA	University of East Anglia
VAT	Virtual Adversarial Training

List of Symbols

K	Total amount of classes, Eq. (2.1)
y_h	Original one-hot label, Eq. (2.1)
α	The extent of softening, Eq. (2.1)
D	Discriminator, Eq. (3.1)
G	Generator, Eq. (3.1)
z	Random noise, Eq. (3.1)
k	Total amount of classes, Eq. (3.2)
n	Total amount of confusion matrices, Eq. (3.2)
c_x	Total amount of true positives for a particular run, x, Eq. (3.2)
k	Total amount of classes, Eq. (3.3)
y_i	One-hot encoded vector with true class labels for a particular class, i, Eq. (3.3)
p_i	Predicted probability for a particular class, i, Eq. (3.3)
k	Total amount of classes, Eq. (3.4)
w_i	Class weight for a particular class, i, Eq. (3.4)
y_i	One-hot encoded vector with true class labels for a particular class, i, Eq. (3.4)
p_i	Predicted probability for a particular class, i, Eq. (3.4)
$N_{S \cup T}$	Total amount of training samples in S and T domains, Eq. (4.1)
S	Source domain, Eq. (4.1)
T	Target domain, Eq. (4.1)
L_*^i	Cross entropy error on Discriminators' predictions, Eq. (4.1)
λ^x	Weighing for a particular module, x, Eq. (4.1)

$N_{S \cup T}$	Total amount of training samples in S and T domains, Eq. (4.2)
λ^{ae}	Weighing for attentive entropy loss, Eq. (4.2)
H	Entropy function, Eq. (4.2)
n	Total amount of training samples, Eq. (4.3)
p_i	Probability of a particular event, i, Eq. (4.3)
N_S	Total amount of training samples in S domain, Eq. (4.4)
S	Source domain, i, Eq. (4.4)
y_i	Ground truth label of a particular sample, i, Eq. (4.4)
$\hat{y}_i$	Prediction for a particular sample, i, Eq. (4.4)
L_d	Domain loss, Eq. (4.5)
L_{ae}	Attentive entropy loss, Eq. (4.5)
L_{cls}	Classification loss, Eq. (4.5)
$\hat{X}_i$	General-purpose features for a particular frame, i, Eq. (4.6)
G_{sf}	Spatial module, Eq. (4.6)
G_{tf}	Temporal module, Eq. (4.6)
G_y	Classification layer, Eq. (4.6)
$N_{S \cup T}$	Total amount of training samples in S and T domains, Eq. (4.7)
λ^{vat}	Weighing for virtual adversarial training loss, Eq. (4.7)
V	Raw video, Eq. (4.7)
r	Adversarial noise, Eq. (4.8)
D_{KL}	Kullback–Leibler divergence, Eq. (4.8)
V	Raw video, Eq. (4.8)
G_t	Spatial module, temporal module, and classification layer put together, Eq. (4.8)
G_r	General-purpose feature extractor, Eq. (4.8)
r	Adversarial noise, Eq. (4.9)
D_{KL}	Kullback–Leibler divergence, Eq. (4.9)
G_t	Spatial module, temporal module, and classification layer put together, Eq. (4.9)

$\hat{X}$	General-purpose features, Eq. (4.9)
$N_{S \cup T}$	Total amount of training samples in S and T domains, Eq. (4.10)
$\hat{X}$	General-purpose features, Eq. (4.10)
$\lambda^{vat} X$	Weighing for virtual adversarial training loss, Eq. (4.10)
μ	Mean, Eq. (4.11)
σ	Standard deviation, Eq. (4.11)
$\hat{X}$	General-purpose features, Eq. (4.11)
$\tilde{x}$	Mixed video, Eq. (5.1)
x	Original, unaltered video, Eq. (5.1)
n	Total number of frames, Eq. (5.1)
λ	Mixing ratio, Eq. (5.1)
t	Frame sequence number, Eq. (5.1)
n	Total number of frames, Eq. (5.2)
t	Frame sequence number, Eq. (5.2)
M	Binary mask region, Eq. (5.3)
x	Original, unaltered video, Eq. (5.3)
$\hat{x}$	Video with randomly shuffled frames, Eq. (5.3)
t	Frame sequence number, Eq. (5.3)
$\odot$	Element-wise multiplication, Eq. (5.3)
W	Frame width, Eq. (5.4)
H	Frame height, Eq. (5.4)
U	Uniform distribution, Eq. (5.4)
α	Beta distribution parameter, Eq. (5.4)
r_w	Bounding box width, Eq. (5.4)
r_h	Bounding box height, Eq. (5.4)

1 Introduction

Due to technology advancements, the cost of data storage and camera devices has been rapidly decaying over the last two decades. As a result, the amount of pictures taken and videos recorded on a day-to-day basis has been increasing exponentially, both in commercial and personal use. Consequently, the need for visual data processing and analysis grew as well, sprouting a number of research areas in computer vision.

Given the abundance of imagery, manual data examination and processing is rather unfeasible in most modern scenarios. Therefore, automated approaches such as deep convolutional neural networks have become the standard choice for a variety of machine learning tasks, thanks to their ability to learn task-specific features in an unmanned manner. However, although CNNs benefit from large amounts of available imagery, their strong performance comes on condition that the data is both task-specific and annotated. This is one the biggest drawbacks of said algorithms - data selection and labelling are often expensive, time-consuming, and tedious processes that require domain knowledge. For example, collecting data for coastal habitat mapping via aerial imagery might require a specific period of the year, specific weather conditions, expensive equipment, pilot skills, and data post-processing, such as keypoint matching for overlapping images to create a sample. Then, to build the dataset, a set of domain experts would need to spend countless hours to manually annotate the imagery and review any disagreements in labels. This describes the bottleneck that most autonomous applications face these days - in order to satisfy the specificity requirement, data has to be collected and labelled every time, and the combined cost of the two creates a barrier for the development of automated image analysis systems.

The above motivates the development of methods that allow the modern algorithms to either learn from available unlabelled imagery or use the already labelled data in more efficient

ways. This thesis explores both, presenting techniques that aim to improve performance of deep learning models in scenarios where additional data collection is not an option.

1.1 Project and Thesis Description

Human enterprise often suffers from direct negative effects caused by substantial growths in population of jellyfish. In 2014, a consortium of Cefas Technology Limited, University of East Anglia, Cefas and Électricité de France Energy proposed JellyMonitor to lower the associated damage and costs. The aim was to build an automated remote sensing tool that would analyse footage acquired by an acoustic imaging device underwater and provide warnings of jellyfish blooms to onshore in advance. The project went through several iterations of trials and deployments, collecting new data each time, which later would be annotated. The deployments would usually last around 2 months and be performed around August, since warmer waters often favour jellyfish growth. However, although jellyfish numbers are increasing worldwide, it is still very hard to predict whether a bloom will occur in a particular area. This makes capturing large amounts of relevant sonar data difficult, which was confirmed by subsequent deployments done by Cefas. Therefore, relying on the data collection process alone was rather impractical and costly.

In October 2018, UEA launched a PhD program as part of the NEXUSS centre for doctoral training, in collaboration with Cefas Technology Limited and Cefas, aiming to develop computer vision and deep learning algorithms that would help JellyMonitor to overcome the aforementioned data issue. This thesis was originally meant to be dedicated to this project, however, in December 2020, Cefas Technology Limited directors had decided to cease the development of the system for reasons independent of University of East Anglia. To mitigate for the unforeseen change, the decision was made to continue with the research plan, however, develop further algorithms for publicly available datasets instead, as data associated with the JellyMonitor project would no longer be collected. Other than that,

the story remains cohesive, and each contribution chapter is a continuation of the previous one.

1.2 Challenges and Solutions

By October 2018, a significant amount of work had been done on the system, putting in place algorithms to convert underwater imagery collected with a high frequency sonar to the Cartesian coordinate system, denoise, remove background, detect, track, and classify objects (French et al., 2018). During the project we established several areas that had to be addressed:

1. The lack of comprehensive and thorough evaluation of JellyMonitor in various settings;
2. Data imbalance, in particular, a small amount of jellyfish present in the collected datasets;
3. A model to classify series of frames that belong to the same object. The one in place classifies each frame individually and averages the outputs, not taking anything else into consideration;
4. Insufficient generalisation, i.e. inability to perform well in new environments.

The first phase of the project focused on putting JellyMonitor through different evaluation scenarios to get a better idea of where the weak points are. During one of the experiments, model generalisation was found as the main limitation of the entire framework. Data generation was suggested to target the specific issue, as well as techniques that were aimed at enhancing the overall performance of the system - an event classifier network, weighted loss, and confidence threshold. The proposed approaches led to an improvement, however, at this stage of the project, it was apparent that JellyMonitor could benefit more from other methods, such as video classification models, domain adaptation, data augmentation, and network generalisation. However, as the project was ceased shortly after, we made

the decision to explore aforementioned topics while working on publicly available datasets instead.

1.3 Contributions and Thesis Outline

The rest of the thesis is described below.

Chapter 2 presents a comprehensive survey of the background literature that covers a general introduction to Deep Learning, systems that are similar to JellyMonitor, and areas of Computer Vision and Deep Learning which were explored during the research period, such as Generative Adversarial Networks, Domain Adaptation, Video Recognition, and various augmentation and regularisation techniques. The order of the topics introduced in the survey is maintained throughout the thesis, meaning that the literature review relevant to Chapter 3 goes first, followed by Chapter 4, with Chapter 5 being the last.

Chapter 3 introduces the JellyMonitor system, describing hardware, datasets and their acquisition, the image processing pipeline, early experiments, and results. During one of the experiments, we find that the classifier network suffers from poor generalisation and does not perform well in a cross validation setting, i.e. new underwater environments. Then, a set of enhancements is proposed to deal with the generalisation issue, leading to a boost in performance and better robustness. The work was published as "Improving Automated Sonar Video Analysis to Notify About Jellyfish Blooms", by Gorpincenko, French, Knight, *et al.* in the IEEE Sensors Journal, 2021 (Gorpincenko et al., 2021a).

In Chapter 4 we discuss the application of a regularisation technique called Virtual Adversarial Training in feature space. Although the idea can be applied in many Deep Learning areas, we chose Video Domain Adaptation as the venue, as that was one of the topics mentioned in the previous chapter's future work. Domain Adaptation is a field within transfer learning that aims to use the knowledge obtained from training a model on one dataset to perform well on the other. At the time, the proposed technique resulted in the new state of the art results in Unsupervised Video DA. While running the experiments, we noticed

that public datasets were rather small and not challenging enough for the modern methods anymore - our system has achieved 98% accuracy or higher on 3 out of 6 adaptation paths, i.e. knowledge transfers from one domain to another. Therefore, we also proposed two new adaptation paths formed by two large and publicly available datasets, which contained more categories and offered greater domain discrepancy when compared to the existing adaptation paths at the time. The paper describing the algorithm was published as "Virtual Adversarial Training in Feature Space to Improve Unsupervised Video Domain Adaptation", by Gorpincenko, French, and Mackiewicz at the Electronic Imaging Conference, 2021 (Gorpincenko et al., 2021b). The dataset paper was published as "SVW-UCF Dataset for Video Domain Adaptation", by Gorpincenko and Mackiewicz, at the International Conference on Image Processing and Vision Engineering - IMPROVE, 2021 (Gorpincenko and Mackiewicz, 2021).

Chapter 5 explores the idea of augmentations for video data further, this time taking the temporal domain into consideration, as suggested at the end of Chapter 4. It introduces a few novel single video transformations, a new way of manipulating augmentation magnitude over time, and the dynamic bounding box and mixing ratio for cut-and-paste and blend algorithms, respectively, resulting in a large number of novel regularisation techniques for videos. The proposed methods exceed the performance achieved by prior art. The work was published as "Extending Temporal Data Augmentation for Video Recognition", by Gorpincenko and Mackiewicz, at the The International Conference on Image and Vision Computing New Zealand, 2022 (Gorpincenko and Mackiewicz, 2022).

Chapter 6 concludes the thesis and discusses potential avenues for future work.

2 Background

The continuous development of deep learning algorithms and image analysis applications results in a significant amount of always evolving literature that one might find helpful when working on a particular sub-topic. This Chapter covers the research that we consider the most useful and relevant to the thesis' contributions. Please note that while we tried to keep it up to date even after publishing our work, some of the most recent publications might be missing, due to the nature of such a fast-paced field.

2.1 Applications in Underwater Monitoring

Underwater monitoring is a research field that includes many sub-topics and is comprised of a large number of applications. For the JellyMonitor purposes, we consider 3 key components - underwater jellyfish detection, sonar use in monitoring applications, and underwater abundance estimation.

2.1.1 Jellyfish

Systems that aim to detect jellyfish mostly use traditional cameras (Martin-Abadal et al., 2020). Whereas optical instruments can provide high resolution RGB images and are effective at short distances, they are unable to produce good underwater imagery at deep water levels and at night without sufficient illumination. There were attempts to capture data with cameras being above the water level, i.e., mounted on a wharf (Llewellyn et al., 2016; Zhou et al., 2015) and built into an unmanned aerial vehicle (Kim et al., 2015b, 2016). In the first case, two 100W floodlights were required to brighten the area of interest, while data was gathered from very shallow waters (5m - 10m); the latter could analyse the water surface only. It is also unknown how robust these approaches are to reflections that occur

in sunny weather. A number of studies attempted underwater imagery collection and analysis (Edgington et al., 2006; Kim et al., 2015a; Rife and Rock, 2003; Spampinato et al., 2008; Walther et al., 2004), however, did not gather enough data to adequately evaluate the performance, hence it is unknown how they would process long-term observations. Some systems did not consider real life conditions and were tested on synthetic data (Matsuura and Fujisawa, 2007). Ultrasound transducers were explored for monitoring purposes, however, the set up included well lit tanks, where the presence of noise is minimal (Vasile et al., 2016). The idea of slicing jellyfish by cutting nets or blades showed its effectiveness in decreasing their abundance (Kim et al., 2016; Park et al., 2012), however, it is rather a short-term solution, as upon being attacked, medusae release large amounts of eggs and sperm that later form planulas, thus amplifying population increase in the future.

2.1.2 Sonars

Multibeam forward-looking sonars offer a good solution to the problem, as they are capable of producing high-resolution images in environments where traditional cameras cannot perform well due to the lack of illumination. They became the common tool for underwater monitoring applications, often used for species identification (Able et al., 2014; Jones et al., 2021; Bothmann et al., 2016; Langkau et al., 2012), abundance estimation (Hughes et al., 2018; Braga et al., 2022; Hayes et al., 2015; van Hal et al., 2017), and behaviour analysis (Helminen and Linnansaari, 2021; Becker et al., 2013; Bennett et al., 2020; Lagarde et al., 2021). However, the aforementioned studies were all done on fish. Identifying, tracking, and counting jellyfish presents additional obstacles, as often medusae might not have such a unique shape, can appear partially transparent on the sonar imagery, and follow different behaviour patterns than fish. All of the above makes it rather easy for monitoring systems to confuse jellyfish with seaweed (French et al., 2018), which in turn leads to increased error rate when it comes to classification and abundance estimation.

2.1.3 Species counting

Manual counting is still the most common approach when it comes to underwater abundance estimation, due to its high accuracy (Wei et al., 2022). However, it is rather a time-consuming and tedious task that requires observers with the relevant domain knowledge (Keefer et al., 2017; Magowan et al., 2012; Grote et al., 2014), which is not a feasible approach for JellyMonitor. Counting individual fish tracks in the automated manner by using a Kalman filter coupled with the nearest neighbour algorithm showed an impressive counting error of less than 11% (Jing et al., 2016, 2018), while the Alpha-Beta algorithm achieved the mean error of 7.2% (Shen et al., 2020). The use of deep learning networks for counting species underwater is still rather new, with the existing studies focusing on scenarios where the presence of species other than the one of interest is unlikely (Liu et al., 2018b; Tarling et al., 2022).

To our knowledge, JellyMonitor, which we will discuss in Chapter 3, is the only system so far that aims to estimate jellyfish abundance in the unmanned manner and in the presence of other underwater phenomena.

2.2 Machine Learning

In this Section, we first cover the most commonly used traditional approaches in computer vision. Then, we present an introduction to deep learning.

2.2.1 Traditional feature learning

Traditional machine learning methods rely on rich hand-crafted feature descriptors that serve as the input to a classifier. Most commonly used feature extraction techniques include the scale-invariant feature transform (Lowe, 2004), histogram of oriented gradients (Dalal and Triggs, 2005), and speeded-up robust features (Bay et al., 2008), whereas the k-nearest neighbours (Cover and Hart, 1967), support vector machines (Boser et al., 1992), decision trees (Breiman et al., 1984) and hidden Markov models (Rabiner and Juang, 1986) are

usually used for classification. Such a framework showed great success in the field for a long time, helping researchers to achieve breakthroughs in various computer vision tasks, such as panorama stitching (Brown and Lowe, 2003), human detection (Zhu et al., 2006), image classification (Vinyals et al., 2012), and object detection (Viola and Jones, 2001). However, the described approach often requires task-specific knowledge and careful descriptor design, which can be costly and time-consuming. Deep learning sometimes can solve this issue by learning features automatically, in the unmanned manner.

2.2.2 Deep learning

Multilayer perceptron

In the early days of neural networks, most DNN architectures used only fully-connected layers as learnable parameters, whereas a set of three or more FCs would form a traditional MLP (Rumelhart et al., 1986). By employing a combination of backpropagation and activation functions Murtagh (1991), multilayer perceptrons became a good solution for data that is not linearly separable (Cybenko, 1989). Being mostly independent from prior knowledge and human effort when it comes to system design is what made them so advantageous over traditional algorithms.

Although MLPs can be used for feature extraction, it is rather inefficient to scale and train, as the neurons in adjacent layers are connected to each other, forming a large number of learnable weights. Moreover, a big portion of spatial information is lost when the image is converted from a matrix to a vector to satisfy the requirements of the input layer. Therefore, it is common to use hand-crafted descriptors as the input, obtained by using traditional feature extraction methods. With the appearance of convolutional layers that do not impose such a high computational cost, fully-connected layers became the standard way of classifying their output, by being placed at the rear of a CNN.

Convolutional neural network

Convolutional neural networks (LeCun et al., 1999) are a family of DNN algorithms (Ivakhnenko, 1971) that became the most popular tool for image processing over the past decade. The main difference between CNNs and MLPs is the presence of convolutions - they are applied to data by sliding learnable kernels over the inputs to produce feature maps, which are then concatenated along the depth dimension to produce the output of a given layer. The process can be repeated many times, by applying subsequent convolutional layers to intermediate features. To perform the classification step and produce the final prediction, feature maps are usually fed into one or more fully-connected layers. Both feature extraction and classification layers are often trained together, combining what used to be the two-step process into one. Since their appearance, they have been extensively used in image classification (LeCun et al., 1999; Krizhevsky et al., 2012; He et al., 2016), image generation (Goodfellow et al., 2014a; Radford et al., 2016; Karras et al., 2019), semantic image segmentation (Long et al., 2015; Chen et al., 2017; Ding et al., 2019), object detection (He et al., 2015; Girshick, 2015; Redmon et al., 2016), depth estimation (Eigen et al., 2014; Zhou et al., 2017; Chen et al., 2019b), and many other areas to achieve state of the art results. An example CNN architecture is shown in Figure 2.1. The rest of this Chapter describes different convolutional neural network variants, their applications, and techniques that aim to enhance models' performance in more detail.

2.3 Image Classification

LeNet (LeCun et al., 1999) is widely recognised as the first convolutional neural network. A rather simplistic architecture, featuring 5×5 convolutional layers, 2×2 average pooling layers, tanh activation functions, and FC layers placed at the rear of the model, was used for the handwritten digit recognition task (Deng, 2012). All LeNet variations significantly outperformed their conventional counterparts in either training speed, test set accuracy, or memory required. By employing boosting (Drucker et al., 1992), LeNet-4 achieved the error

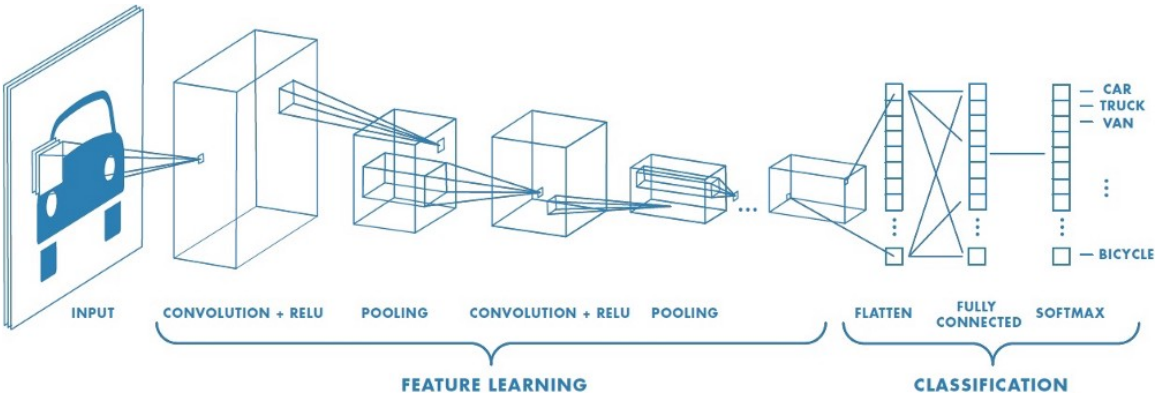


Figure 2.1: Convolutional neural network architecture. Figure from `https://bit.ly/2HGwg9N`.

rate of 0.7%, the lowest of all. When it comes to the traditional approaches, the closest score of 1.1% was attained by the Reduced Set Support Vector Technique (Burges and Schölkopf, 1996), which is about 60% more computationally expensive than the proposed CNN. However, the scalability of deep learning models was still a concern - for many, it was unfeasible to train deeper networks for more challenging problems that involved three colour channels, higher resolution images, larger training sets, or an increased number of classes (Krizhevsky et al., 2009; Coates et al., 2011; Netzer et al., 2011; Russakovsky et al., 2015). Moreover, the tasks themselves started becoming more complicated.

The circumstances changed when researchers started training neural networks on GPUs. Their ability to perform many more simple instructions in parallel when compared to CPUs makes them the superior processing unit in deep learning. Krizhevsky et al. (2012) introduced AlexNet - a network with 8 layers and 60M parameters, which is a thousand times more than LeNet. AlexNet had significantly more channels, utilised kernels of different sizes, as well as dropout (Srivastava et al., 2014), and swapped average pooling and tanh activation functions with max pooling and rectified linear unit (Fukushima, 1975), respectively. The model won the 2012 version of the ImageNet competition (Russakovsky et al., 2015), which is the most commonly used benchmark for evaluating image classifiers to this day. AlexNet attained the top-5 error rate of 16.4%, whereas the classic techniques could

not break the 25% barrier. Shortly after, the authors of ZFNet (Zeiler and Fergus, 2014) highlighted the importance of understanding of how CNNs work internally and introduced deconvolution - a way to convert feature maps into a human-comprehensible visual form. By using the same architecture as AlexNet and only changing hyperparameters, Zeiler and Fergus (2014) managed to decrease the error further down to 11.7%, which made ZFNet the winner of ImageNet 2013.

Simonyan and Zisserman (2014b) introduced the VGG architecture, which marks the beginning of very deep CNNs. They argued that having more layers improves one's ability to be able to fit complex, non-linear data, and hence depth consideration is an important part of the model design. The architecture was extended all the way to 19 layers and achieved the top-5 error rate of 7.3% on ImageNet, winning the challenge in 2014 in the localisation setting. Unlike previous work, VGG utilised only 3×3 convolutions. Multiple stacked convolutions are capable of replicating bigger receptive fields, however, an additional benefit comes from the non-linear activation functions placed in between. The most commonly used version, VGG-16, consists of 138M parameters. JellyMonitor, which we will discuss in Chapter 3, uses a VGG-based architecture to classify sonar frames. At the same time, GoogLeNet (Szegedy et al., 2015) was introduced, which supported the idea of having much deeper models, however, the main objective was to reduce the parameter count and memory usage. It went up to 22 layers, but replaced fully-connected layers with global average pooling and 1×1 convolutions, which significantly reduces data dimensionality. Szegedy et al. (2015) also introduced the inception module that allows the use of multiple filter sizes in a single layer, making the network not just deeper, but wider as well. Despite layers with multiple kernels, the total number of parameters was only 5M. GoogLeNet became the ImageNet classification winner in 2014, with 6.7% top-5 error.

The architectures grew larger, and the problem of vanishing gradients came to light. As the signal goes through every component of a neural network before reaching the initial layer, the risk of it having too little impact on the weights or fading completely increases. In the

worst case scenario, this can stop a model from training completely (Basodi et al., 2020). GoogLeNet (Szegedy et al., 2015) tackles this issue by connecting intermediate layers to auxiliary classifiers and thus making sure that no layer is too far away from the output. The VGG architectures (Simonyan and Zisserman, 2014b) were split into two parts and trained separately to avoid the problem. Pascanu et al. (2013) proposed a regularisation term that forces the error signal not to vanish as it travels back to the input layer. Batch normalisation (Ioffe and Szegedy, 2015) re-centers and re-scales the data throughout a network. ResNet (He et al., 2016) introduced residual or shortcut connections that allowed the gradient to skip intermediate layers on their way back to the head of a model. The latter allowed the authors to train the deepest CNNs to date without suffering from vanishing gradients, going up to 152 layers. ResNet took the first place in the ImageNet 2015 classification competition, with 3.57% top-5 error. Nowadays, it is a common practice to have either batch normalisation or residual connections in CNN architectures.

The top-5 error was reduced to 3% in 2016 by Xie et al. (2017), who proposed to expand the dimensionality of residual blocks, up to a total of 32 paths, whereas the original used just one. The squeeze-and-excitation block (Hu et al., 2018), which adaptively recalibrates channel-wise feature responses, served as the foundation for the framework that won the ImageNet 2017 classification challenge, with 2.25% top-5 error. The most recent architectures advanced so much that it became rather usual to score 1% error in the top-5 setting (Yuan et al., 2021; Pham et al., 2021). Therefore, the attention has shifted to the top-1 accuracy. At the time of writing, the state of the art models were approaching 91% (Dai et al., 2021; Wortsman et al., 2022; Yu et al., 2022; Chen et al., 2022).

2.4 Generative Adversarial Networks

Generative adversarial networks (Goodfellow et al., 2014a) are a modern method that views generative modelling as a game between two models called a generator and a discriminator. The aim of the discriminator is to be able to distinguish between real and generated samples,

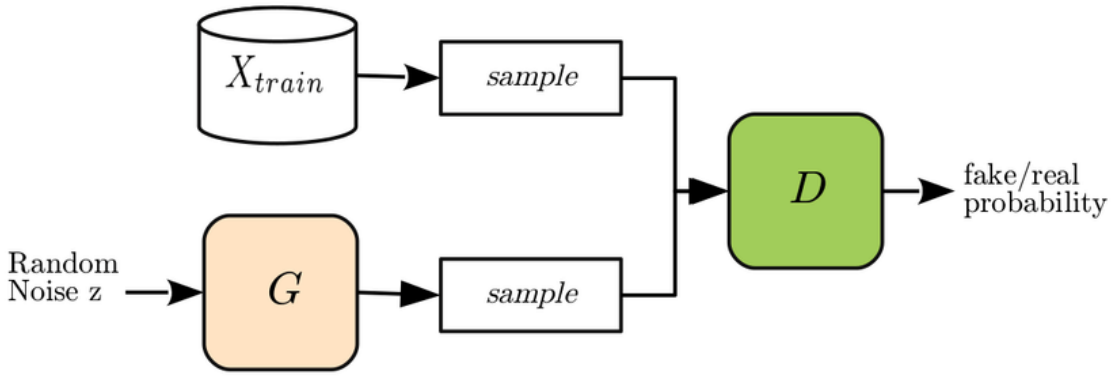


Figure 2.2: The first generative adversarial network architecture (Goodfellow et al., 2014a). Figure from (Hayes et al., 2017).

whereas the generator learns to produce data that tricks the discriminator into thinking that it is real. That results in a two-player minimax game, where improving the performance of one network negatively impacts the performance of the other. An example GAN architecture is shown in Figure 2.2. Despite the fact that a large amount of work has been done on generative adversarial networks and recent methods have been able to produce samples that often are indistinguishable from real data even for human observers (Karras et al., 2017, 2019; Brock et al., 2018), they are still hard to train due to their unstable nature (Arjovsky et al., 2017; Gulrajani et al., 2017; Salimans et al., 2016; Metz et al., 2016)

The first version of GAN (Goodfellow et al., 2014a) used multilayer perceptrons for both networks, which restricts the ability to scale well to large applications. Radford et al. (2016) proposed DCGANs, where models are made of convolutional layers instead. The generator slowly grows in size, expanding the input noise vector to a full resolution image, while the discriminator does the opposite, reminiscent of the conventional CNN architecture. Coupled with batch normalisation (Ioffe and Szegedy, 2015), DCGANs greatly improved the stability of the training process.

Although GANs are capable of creating realistic looking samples, there is very little control over what modes are generated. Conditional GAN (Mirza and Osindero, 2014) changes that by providing label information to both networks as an additional input. With time, the

discriminator learns how different classes from the real data distribution look by extracting features related to a particular task, which gives guidance to the generator on how to produce specific categories. Information maximising GAN (Chen et al., 2016), on the other hand, does not require labels. Instead, it uses latent codes as an additional input, which the system learns by itself, in an unsupervised fashion.

A common problem in GAN training process is mode collapse - a situation when all generated instances converge to one point, regardless of the input (Metz et al., 2016). The goal of the generator is to fool the discriminator, and ideally it would do so by learning the real data distribution. In reality, however, it is much easier for it to find a point that the discriminator would consider real. If at any stage the discriminator learns that the point is fake, the generator can move its outputs to another point, resulting in the game of cat-and-mouse. Minibatch discrimination (Salimans et al., 2016) tackles the problem by telling the discriminator how statistically close samples are to each other in a batch. Progressive GAN (Karras et al., 2017) simplifies this idea by getting rid of trainable parameters and computing standard deviation for each feature in a batch instead. Techniques that were initially introduced to solve other issues, such as spectral normalisation (Zhang et al., 2019) and R1 regularisation (Mescheder et al., 2018), have been proven to reduce the risk of mode collapse as well.

Goodfellow et al. (2014a) proposed a minimax two-player game framework, where the models are trained concurrently to find a Nash equilibrium in a non-cooperative game. Each network updates itself with no respect to another player in the game, and updating gradient simultaneously cannot guarantee convergence. Moreover, it requires both models performing at the same level - if the discriminator is too good, the gradient of the loss function falls to zero, and the training process becomes slow or stuck; if the discriminator is too bad, the generator does not have enough guidance to improve itself. WGAN (Arjovsky et al., 2017) uses Wasserstein distance - a measure of the distance between two probability distributions. By applying a transformation on the formula, the authors propose a new

formulation - now it can be seen as learning a K-Lipschitz continuous function that aims to minimise the gap between two distributions instead of the minimax game. Arjovsky et al. (2017) proposed clipping the discriminator's weights to ensure its smoothness. Gulrajani et al. (2017) proposed gradient penalty, which provides a compact parameter space without losing some parts of the gradient. The idea was then further improved by penalising the gradient on real data alone (Mescheder et al., 2018), and was used to build the state of the art systems at the time, such as StyleGAN (Karras et al., 2019) and BigGAN (Brock et al., 2018).

Generative adversarial networks are very good at generating realistic textures, such as water, sky, grass, and fur. However, they struggle to find dependencies across image regions. Therefore, structural patterns are often lost - an animal might not have a head, or is drawn without clearly defined face. To combat this, Zhang et al. (2019) proposed an attention module that calculates response at a position as a weighted sum of the features at all positions. ProGAN (Karras et al., 2017) tackles the problem by progressively increasing the resolution of generated images - the idea is to add new layers to both models only when they perform well on the smaller scale, where dependencies across regions are positioned closer to each other. This approach also makes training much faster and stabilises the process, as both networks start from rather simple tasks.

StyleGAN (Karras et al., 2019) combines the progressive architecture with neural style transfer. Instead of passing the noise vector directly to the generator, it first goes through a learnable mapping network, which results in a style vector. The vector is then fed to each block of the generator model through adaptive instance normalisation (Huang and Belongie, 2017). Since each block is responsible for a different resolution, the generator learns to associate the style information with features that are present at various scales, such as freckles or hair at higher resolutions and pose or identity at the lower ones. By incorporating mixing regularisation, Karras et al. (2019) managed to achieve much greater control over synthesised images. StyleGAN has seen several improvements over the last few

years (Karras et al., 2020a,b, 2021), and is arguably the most popular generative adversarial network to this day. In Chapter 3, we will use StyleGAN to artificially expand the training set, which helps us to achieve higher frame accuracy.

2.4.1 Lack of data

In the realm of machine learning, the availability of diverse data is often regarded as the foundation for developing robust and accurate models. In practical scenarios, however, finding high-quality, labeled, and diverse datasets is a common challenge. To address this problem, one pivotal solution has emerged in the form of GANs - the ability to generate data that is virtually indistinguishable from real data allows to augment and expand existing datasets. By enriching available training data, GANs help to enhance the classifiers' capacity to generalise and make more accurate predictions.

Generative adversarial networks are particularly useful when it comes to imbalanced datasets, where certain classes might be underrepresented, therefore leading to biased model outputs. GANs excel in those situations, as the engineer has control over what categories are being generated, which helps with balancing the problematic dataset (Mirza and Osindero, 2014). This balance improves the model's ability to recognise and classify rare instances, making it particularly suitable for applications where collecting those minority classes is difficult, such as medical applications or JellyMonitor, which we introduced in Chapter 1. In Chapter 3, we will apply StyleGAN (Karras et al., 2019) to enhance the JellyMonitor's classifier and perform a number of experiments to find the most optimal strategy for generating fake samples for an imbalanced dataset.

GANs also offer a unique property of extending data distribution into unexplored regions of pixel or feature space. In cases where the real data is limited to a specific domain or cluster, generative adversarial networks can generate samples in different regions, expanding the underlying data distribution, which allows the classifier to generalise better (Isola et al., 2017). GANs can generate synthetic data that preserves the statistical properties of the

original data, which safeguards sensitive or private information. This feature enables the development of models without compromising privacy, making it a strong tool in scenarios where security is a concern (Wu et al., 2018). When the training data is scarce, GANs can be employed to to pretrain models, resulting in more effective initialisation (Brock et al., 2018). Generative adversarial networks excel in filling missing values in datasets, ensuring that machine learning models have enough information about the entire distribution of interest (Pu et al., 2016).

2.4.2 Generative adversarial network evaluation

In order to evaluate the quality of generated imagery, one has to be very familiar with the train set and investigate a huge number of samples, which is subjective, tedious, and often infeasible.

When it comes to conditional image generation, the generator model can be used to produce the train set for an image classification task. If a classifier CNN is trained on fake instances and tested on the real ones, the final test accuracy can serve as a score for that particular generator (Isola et al., 2017; Radford et al., 2015). However, this approach relies on the quality of the classifier network, gives little to no information about the variety of generated samples, and is not applicable to unsupervised image generation or cases where only one class is considered.

Maximum mean discrepancy (Gretton et al., 2012) independently draws samples from fake and real distributions and measures the dissimilarity between the two. Reconstruction error (Xiang and Li, 2017) evaluates the difference between a test image and its closest generated image. Nearest neighbours (Cover and Hart, 1967) in feature and pixel spaces can be used to see whether the generator has memorised the training set, mainly to detect overfitting.

Inception score (Salimans et al., 2016) uses the pre-trained Inception network (Szegedy et al., 2016a) to evaluate the quality and variety of generated images. The motivation is

that images with meaningful objects should have high confidence for one class and low for the rest. On top of that, overall distribution of predicted classes should be balanced, which indicates a diverse set. However, the score is highly dependant on the training set that was used for the Inception model - if a generated category is not present there, most likely the model will not be confident about its predictions. Moreover, this approach requires the train set for the classifier to come from the same visual domain as the one used for training the GAN.

Fréchet inception distance (Heusel et al., 2017) measures the similarity between two sets of images by comparing their statistics. Similar to inception score, it uses the pre-trained Inception model, this time to extract features from an intermediate layer. Since FID operates on the feature level, it is less dependant on the initial training set that was used for the classifier. However, it lacks the ability to penalise for having the same samples in both datasets. Therefore, the nearest neighbours and FID is the most commonly used combination when it comes to GAN evaluation (Brock et al., 2018; Karras et al., 2017, 2019).

2.5 Domain Adaptation

As long as there is plenty of imagery and both training and test splits come from similar distributions, modern neural networks generally show very good results. However, when the splits exhibit different appearance (e.g.: scenery collected in a video game to train a segmentation network which is then used on the real world landscapes), their performance drops significantly. Such visual discrepancy is often referred to as domain shift, or domain gap (Vázquez et al., 2012; Stark et al., 2010; Sun and Saenko, 2014; Liebelt and Schmid, 2010). Collecting and annotating new data from the domain of interest might be tedious, expensive, or even impossible. Domain adaptation aims to solve this issue by exploring algorithms that learn robust classifiers in the presence of a shift between train (source domain) and test data (target domain). The majority of work in the field is focused on the unsupervised

case, where the annotated data is available only in the source domain, although extending these frameworks to semi-supervised cases is generally rather straightforward.

2.5.1 Shallow domain adaptation

Gopalan et al. (2011) estimate the gap by viewing the domains on the Grassman manifold and sampling points along the geodesic between them. That allows to place the source instances onto the sampled points, enabling the classifier to learn and later distinguish between projected target data. Geodesic flow kernel (Gong et al., 2012b) extends the idea by projecting features too, which allows to learn a kernel function that is then used to extract domain invariant representations. The subspace alignment approach (Fernando et al., 2013) learns a mapping function that projects source points onto the target data Grassman manifold. (Sun and Saenko, 2015) argue that aligning the subspaces alone might not be sufficient for good point alignment, as the sample distributions can have different characteristics. They propose finding a mapping between the source and target data in the subspaces, as well as the subspace bases. Transfer component analysis (Pan et al., 2011) aims to find a feature extraction method that preserves the data properties and yet can be used by both domains via a set of common transfer components that project features onto a common space. Joint distribution adaptation (Long et al., 2013) learns a feature transformation to satisfy joint expectations of both domains when it comes to a particular class. Since the target data is not annotated, they use pseudo labels that are estimated by a classifier trained on the source domain. The landmark-based approaches (Gong et al., 2012a, 2013) state that not all samples are created equally for adaptation. Therefore, they pick subsets of most desirable labelled instances from the source domain to drive the adaptation mechanism. To find data points that are distributed closely to the target domain, all source samples are mapped to the reproducing kernel Hilbert space, where the differences can be compared in sample means (Gretton et al., 2012). Then, adaptation of each subset leads to a new feature representation, which can be later combined to build domain-invariant features.

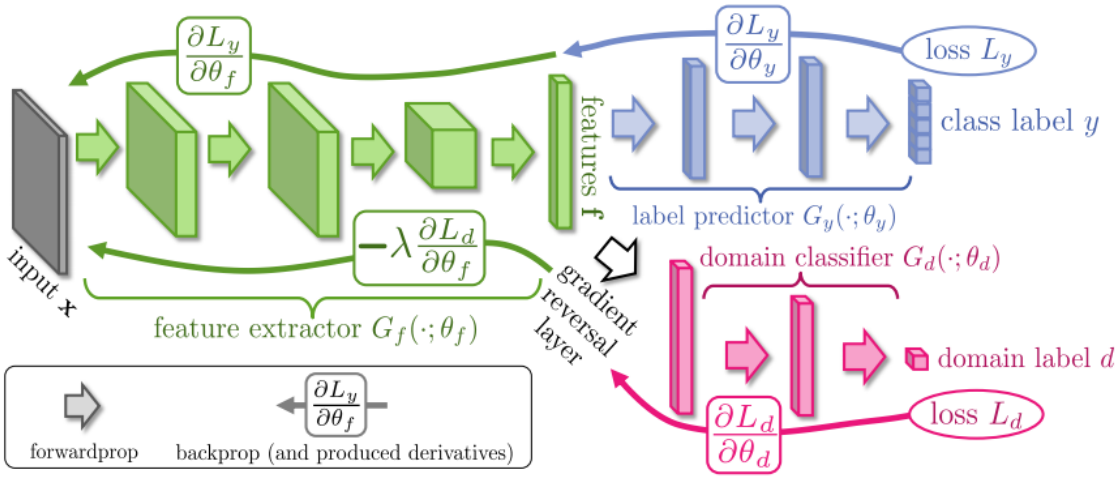


Figure 2.3: Domain adaptation by backpropagation. Figure from (Ganin and Lempitsky, 2015).

2.5.2 Deep domain adaptation

Domain-invariant feature learning

Most recent deep domain adaptation approaches aim to bridge the domain gap by learning a domain-invariant feature extractor. Ganin and Lempitsky (2015) utilise the idea of adversarial training (Goodfellow et al., 2014a), and propose decreasing discrepancy between feature distributions of two domains, similar to how GANs minimise discrepancy between real and synthesised data. This is achieved with the help of a gradient reversal layer and a combination of classification and domain losses, which is demonstrated in Figure 2.3.

Adversarial discriminative domain adaptation (Tzeng et al., 2017) proposes splitting the process into two stages. First, a feature extractor (source CNN) and a classifier are trained in traditional, supervised manner on source data. Next, the source CNN gets frozen and another feature extractor (target CNN) is introduced, which is guided to produce similar features to those of a source CNN by a domain discriminator. Finally, the target CNN and the classifier networks are combined to classify samples from the target domain. Inspired by WGAN (Arjovsky et al., 2017), Shen et al. (2018) view the domain discriminator as a model that approximates Wasserstein distance between the feature distributions. To improve the

training stability further, Wang et al. (2018) apply the gradient penalty (Gulrajani et al., 2017) to the network as well.

Virtual adversarial domain adaptation (Shu et al., 2018) incorporates virtual adversarial training (Miyato et al., 2018) and conditional entropy loss to place the decision boundaries in low-density regions, thus making the model locally-Lipschitz and satisfying the cluster assumption. The authors also propose a refinement training phase with a mean teacher (Tarvainen and Valpola, 2017), where the target domain gets pseudo-labelled by the classifier, constructing a new pseudo-source domain, which iteratively narrows the gap. Mao et al. (2019) note that VAT makes the network smooth only around data points. Instead, they propose using mixup (Zhang et al., 2017), which incorporates the locally-Lipschitz constraint to the areas in-between training instances.

Generate to adapt (Sankaranarayanan et al., 2018a) trains a generative adversarial network to produce images in the source domain while using both source and target features, which encourages the feature extractor to yield domain-invariant representations. Sankaranarayanan et al. (2018b) argue that there is a benefit in reconstructing images in the target domain too, and utilise a similar approach for semantic segmentation.

Domain mapping

Domain mapping performs adaptation through translating instances from one domain to another. Pixel level domain adaptation (Bousmalis et al., 2017) provides source images as well as noise vectors to the generator, which learns to translate the former to the target domain. The task-specific classifier is trained on both source and transformed samples, where the latter attains the labels of the pre-transferred data. Inspired by the cycle-consistency loss (Zhu et al., 2017), Hoffman et al. (2018) adopt the original framework with a few key changes. They propose having an additional discriminator on the feature level, as well as semantic consistency loss to ensure its preservation when mapping source images to the target domain. Symmetric bi-directional adaptive GAN (Russo et al., 2018) employs two

generator networks (from source to target and vice versa), as well as a discriminator and a classifier for each domain. That allows to transform labelled source instances to the target domain and back, while the source classifier provides feedback about changes in semantics. During test time, target images are translated to the source domain, while both classifiers give predictions for their respective versions of the samples. Finally, the outputs are linearly combined to produce the final scores.

Hong et al. (2018) use the GAN framework to operate exclusively in the feature space. They use a conditional generator that learns to transform source features to the target domain, while a deconvolution model converts both original and mapped descriptors to segmentation labels.

Ensembling

Ensemble methods aim to train a set of diverse networks that can be utilised all at once during test time to compensate for each other's shortcomings. Asymmetric tri-training (Saito et al., 2017) trains a single feature extractor and two classifiers on source data while penalising the latter for having similar weights, which ensures a good level of dissimilarity between them. Then, the models are used in ensemble to pseudo-label samples from the target domain and train another classifier. Self-ensembling for domain adaptation (French et al., 2017) uses the student and teacher framework (Tarvainen and Valpola, 2017), where one model is a running average of the other's weights. To ensure high stochasticity between updates in a single network, the method employs aggressive and rich random data augmentation, which makes the model to change the weights rather abruptly. Moreover, French et al. (2017) found that ignoring pseudo-labels with low confidence stabilises training and improves the system's performance. Zou et al. (2018) discover that relying on the prediction confidence favours easy categories and risks completely ignoring the harder ones, resulting in a bias. They propose a class-wise weighting term to balance out the class distribution, as well as a self-training framework that alternates between two stages: optimising the network based on generated pseudo-labels and generating new annotations based on the updated

classifier. The above results in a temporal ensemble (Laine and Aila, 2016), which is formed by one model.

2.5.3 Video domain adaptation

Videos, unlike static images, consist of a series of moving visual frames. This dynamic nature introduces two types of shifts: one in temporal and the other in spatial. Understanding and addressing these shifts is crucial for adapting computer vision models to different domains. In this context, it is essential to consider the role of both the foreground and the background when recognising human actions. A study by Sultani and Saleemi (2014) explores this concept and finds that giving more importance to the person performing actions can enhance the model's ability to work effectively across different domains. In addition to considering foreground and background, researchers have developed various approaches to adapt computer vision models for video data. Xu et al. (2016) introduced a dual many-to-one encoder architecture, which trains networks to map features obtained from traditional computer vision techniques into a common space, facilitating domain adaptation. Another technique, known as action modeling on latent subspaces (Jamal et al., 2018a), views videos as sequences of points on the Grassman manifold. This perspective allows for the use of methods like subspace alignment (Fernando et al., 2013) and the geodesic flow kernel (Gong et al., 2012b) to perform domain adaptation. The aforementioned methods focused on small-scale video-based DA datasets, where the domain gap along the time dimension is rather small. Therefore, the need for precise temporal alignment was limited. Chen et al. (2019a) address this by introducing two new large-scale datasets, in addition to proposing the TA^{3}N model, which encodes the relation between frames into features and aligns local temporal dynamics with larger domain discrepancy through the domain attention mechanism. In Chapter 4, we will use the TA^{3}N framework as a baseline to evaluate the effectiveness of virtual adversarial training in feature space in the unsupervised video domain adaptation setting. This approach allows us to explore how to make computer vision models more adaptable and robust when working with video data from different sources.

2.6 Model Regularisation

Overfitting describes a scenario where a classifier memorises input data and hence fails to generalise and perform well on new, unseen data, even though it comes from a similar distribution. This is usually indicated by a large gap between training and testing errors.

Many methods explored regularising networks by directly looking at their parameters. Weight decay approaches restrict parameters to a compact space or shrink them towards zero, which makes some features less important or obsolete (Wen et al., 2016; Krizhevsky et al., 2012). That gets rid of a lot of bias, as more attention is paid to the features themselves, instead of their unique combinations. Dropout (Srivastava et al., 2014) randomly disables activations in fully-connected layers, which ensures that all the weights are equally important. DropConnect (Wan et al., 2013) generalises dropout by randomly turning off the weights instead. DropAll (Frazao and Alexandre, 2014) combines the two aforementioned approaches to achieve better classification performance. DropBlock (Ghiasi et al., 2018) drops continuous regions in images and feature maps, which forces models to learn in the complete absence of some certain semantic information, such as feet or tails.

Another technique that helps DNNs to avoid having sparse gradients is data normalisation, which compresses the inputs to a common tight space (LeCun et al., 2012). The idea of batch normalisation (Ioffe and Szegedy, 2015) is to apply the operation between the layers as well, which results in reduced covariance shift between the layers and improvements in training speed. Various modifications of batch normalisation have been explored up to date, including having a reference batch (Salimans et al., 2016), or deriving statistics from batches and fusing it with loss functions (Karras et al., 2017). Normalising network weights directly has proven to be effective at reducing training time (Salimans and Kingma, 2016). Layer normalisation (Ba et al., 2016) normalises inputs across their features, which makes the operation independent of batch statistics that can vary a lot at small batch sizes.

Ulyanov et al. (2016) restrict layer normalisation to individual channels, which leads to better stylisation results.

Early stopping (Prechelt, 1998) regularly tests the model on the validation set, which allows to spot overfitting early. Once the validation error stagnates or starts rising, the training process gets stopped. Szegedy et al. (2016b) hypothesise that having large logit gaps make the classifier less adaptive, and therefore penalise the network for being overly confident about its predictions by smoothing the ground truth annotations:

$$y_s = (1 - \alpha) * y_h + \alpha/K, \tag{2.1}$$

where K is the total amount of classes, y_h is the original one-hot label, and $\alpha \in [0, 1]$ determines the extent of softening.

Szegedy et al. (2013) discovered that once trained, many architectures, including state of the art models, are still very vulnerable to perturbations, even when they are so small that human eyes cannot see the difference. To address the problem, they propose adversarial training, which alters samples in the direction to which the classifier's prediction is most sensitive. Such regularisation results in more robust and smoother models, as well as higher test accuracy. Goodfellow et al. (2014b) views adversarial perturbations as cheap, linear approximations, which significantly speeds up the training process without compromising the quality of generated samples or the networks' performance. Pseudo-ensembles (Bachman et al., 2014) utilise random noise for semi-supervised learning by penalising high variance in outputs for perturbed data. Virtual adversarial training (Miyato et al., 2018) replaces stochastic noise with adversarial perturbations that seek directions to which predictions change abruptly. Similar to the pseudo-ensembles approach, VAT does not require ground truth annotations, essentially extending adversarial training to semi-supervised and unsupervised settings. In Chapter 4, we will extend virtual adversarial training to feature space and find that it leads to improvements in the unsupervised video domain adaptation setting.

2.6.1 Data augmentation

Data augmentation expands the training set by altering available data. The idea of applying noise to the input images to regularise artificial neural networks has been around for a while and showed its effectiveness (Oh et al., 1991; Sietsma and Dow, 1991; Reed et al., 1992; Bishop, 1995). This is different to previously discussed adversarial training, as the noise is applied randomly. Nowadays, it is more common to use other basic image modifications, such as axis flipping, rotations, translations, random cropping, and colour space alterations (Krizhevsky et al., 2012; Simard et al., 2003; Cireşan et al., 2011, 2012). These techniques are easy to implement, bear minimum computational overhead and are very likely to preserve semantics after transformation. However, combining the aforementioned operations together can result in heavily inflated datasets and high risk of label warp. Therefore, a number of studies has been done on search algorithms that aim to find the optimal subset of augmentations for a particular task. Given a dataset and a pool of transformations, AutoAugment (Cubuk et al., 2018) utilises a search algorithm to find the best augmentation policy for a particular task and uses validation error as the guide. SmartAugment (Lemley et al., 2017) learns a model that combines two or more samples from the same category to generate new data. DeVries and Taylor (2017a) apply a set of simple transformations to data points in a learned feature space. Finally, RandAugment (Cubuk et al., 2020) presents an efficient framework that works out of the box for applying operations sequentially and without a separate search phase.

Similar to Dropblock (Ghiasi et al., 2018), Cutout (DeVries and Taylor, 2017b) randomly drops rectangular image regions, which prevents the model from developing a bias towards some particular areas or visual features. Instead of setting pixel values to zero, RandErase (Zhong et al., 2020) proposes replacing the erased regions with stochastic noise. Inoue (2018) discovered that performance gains can be observed even by averaging pixel values of two random images and retaining only one out of the two labels. MixUp (Zhang et al., 2017) introduces an improved framework, where samples are blended at different degrees,

as well as their corresponding labels. Following the idea of soft labels for image mixing, CutMix (Yun et al., 2019) improves upon Cutout by replacing the chosen areas with blocks from other samples. CowMask (French et al., 2020) gives the regions Friesian cow-like appearance, further improving semi-supervised results attained by CutMix. CutMixUp (Yoo et al., 2020) combines the properties of MixUp and CutMix, which results in blending cropped areas at various ratios. Summers and Dinneen (2019) reveal a whole range of viable mixing techniques, suggesting that the space of practical augmentation methods is much broader than previously realised.

Although different from conventional data augmentation, creating synthetic data with the help of generative adversarial networks is yet another way to expand a dataset. With recent advancements in the field, GANs are now able to generate images that look real to human observers, in spite of illustrating entities that are not present in the training set (Karras et al., 2019; Brock et al., 2018). The GAN framework also can be extended to improve the quality of samples created by variational auto-encoders (Doersch, 2016) or perform style transfer to map existing imagery to the domain of interest (Zhu et al., 2017; Terayama et al., 2019; Lee et al., 2019).

Video data augmentation

Although a substantial amount of work has been done on spatial augmentation, the field of temporal augmentation remains under-explored. Random Mean Scaling (Kim et al., 2020a) stochastically varies the low-frequency feature components to regularise classifiers, whereas FreqAug (Kim and Ha, 2021) experiments with randomly removing them. Benaim et al. (2020) found that changing video speed during training results in stronger space-time representations that can boost the performance of self-supervised action recognition. Boundary-sensitive video synthesis (Xu et al., 2021) proposes smooth clip blending, which leads to better results in the action localisation setting. RandAugment-T (Kim et al., 2020b) extends the spatial-only RandAugment framework to the time dimension by sampling two magnitude points that are placed at the start and the end of each video. It also presents a set

of modifications on cut-and-paste and blend algorithms, such as CutOut, CutMix, MixUp, and CutMixUp, to produce temporally localisable features. In Chapter 5, we will expand on the work done by Kim et al. (2020b). We will show that some of the proposed techniques can be extended even further to fully utilise the time domain and achieve a deeper level of temporal perturbations, which results in more accurate and robust classifiers.

2.7 Video Recognition

A clear-cut approach to video classification using CNNs is to include the temporal domain by extending the dimensionality of convolutional operations. 3D filters achieved superior results when compared to 2D, proving that the time domain has a lot of value (Ji et al., 2013). The inclusion of the temporal axis opened up a whole research area that is aimed at exploring its various fusion techniques. The most popular ones are slow fusion to improve the time awareness of the model (Karpathy et al., 2014), late fusion, where temporal features are blended at the last layer (Karpathy et al., 2014), longer fusion, which explores the benefit of extending the temporal depth (Varol et al., 2017), and ensembling networks with different temporal awareness (Varol et al., 2017). Finally, a combination of 2D an 1D kernels is proposed to substantially reduce the amount of learnable parameters without any loss in performance (Sun et al., 2015).

Motivated by the fact that humans use different streams to process appearance and motion data, multiple stream models were proposed (Simonyan and Zisserman, 2014a). The aim is to have separate spatial and temporal tracks, hence making it easier to encode relevant features in the respective streams. This is further enhanced by supplying different inputs - whereas the spatial path takes RGB frames, which contain appearance information, the temporal path receives optical flow frames that contain motion data. Later work shows that earlier fusion of the streams allows to retain the performance while halving the amount of learnable parameters (Feichtenhofer et al., 2016). Optical flow has been extensively used in research and is known for providing a significant boost to the recognition performance

(Tran et al., 2015; Wang et al., 2016; Carreira and Zisserman, 2017). However, calculating optical flow in advance is rather a time-consuming task that requires a lot of disk space (Zhang et al., 2020), which makes it inefficient for many modern applications. Optical flow estimation approaches (Dosovitskiy et al., 2015; Ilg et al., 2017; Ranjan and Black, 2017) propose training a separate network to estimate the flow and hence speed up the process, however, that is rather a different branch of research which is separated from the video recognition task.

More recent works suggest using lightweight modules to create short-term motion representations. For example, persistent appearance networks (Zhang et al., 2020) lift the reliance on optical flow and instead focus more on modelling the small displacements of motion boundaries by looking at the difference of the low-level features between the adjacent frames. Temporal modules (Wang et al., 2021) capture both short-term and long-term temporal information in a video into compact features and fuse them with the features produced by the backbone architecture. Event adaptive networks (Tian et al., 2022) utilise the latent motion code module to exploit motion information in the feature space, making the process more efficient and effective.

2.8 Discussion

In this thesis, we address several open problems that have been found during the literature review process.

We discovered that JellyMonitor (French et al., 2018) is the only system that aims to estimate jellyfish abundance in the unmanned manner and in the presence of other underwater phenomena. We propose several improvements to JellyMonitor in Chapter 3, further advancing the system and improving its performance and robustness. One of the main Chapter contributions is the application of StyleGAN (Karras et al., 2019) to a real-world problem - a rather novel use of the generative adversarial networks at the time of writing. Moreover,

we introduce an event classifier network - something that we have not come across in the literature.

We found that despite its popularity, virtual adversarial training (Miyato et al., 2018), has yet to be employed within the realm of feature space. In Chapter 4, we delve into the exploration of this utilisation. We also provide our view and findings on entropy minimisation and iterative refinement training (Shu et al., 2018), further simplifying their application in the unsupervised video domain adaptation setting. Finally, literature review revealed the lack of comprehensive adaptation paths, which we address by introducing two new paths formed by two large and publicly available datasets.

We discovered that the incorporation of the temporal domain into the creation of augmentations for video samples represented a novel concept in the field (Kim et al., 2020b). Our exploration of video data augmentations takes this concept to the next level in Chapter 5, by introducing a number of single sample augmentations, a novel way of manipulating augmentation magnitude over time, and the utilisation of dynamic bounding boxes and mixing ratios within cut-and-paste and blend algorithms.

3 Improving Automated Sonar Video Analysis to Notify About Jellyfish Blooms

Jellyfish population has increased worldwide due to climate change and human activity, such as overfishing, aquaculture, and coastal constructions, that create satisfactory environments for polyp settlement and winter survival (Mills, 2001; Pauly et al., 2009). This phenomenon is not limited to a set of particular species and has been noticed in many different coastal ecosystems (Graham, 2001; Link, 2006; Malej et al., 2012; Shiganova, 1998). Moreover, this growth in numbers is unstable and often unpredictable; thanks to the explosive nature of gelatinous zooplankton life cycles, they usually come in periodic blooms (Mills, 2001; Boero et al., 2016). Such high volumes of jellyfish are known for clogging cooling water intakes at coastal power and desalination plants, filling and splitting fishing nets, killing captured fish, swarming beaches, and stopping ship engines (G Masilamani et al., 2000; Kim et al., 2016; Purcell et al., 2007), which incurs large damage and associated costs to a number of marine-related industries, such as travel, shipping, energy generation, and tourism.

Imaging sonars are a good solution to the problem, as they can produce images at movie-like frame rates in the underwater marine environment, where low-light video systems struggle to provide useful imagery and notify about blooms. With recent advancements in computer vision and deep learning, it is possible to collect, process and analyse sonar data in an unmanned real-time manner. All of the above is performed by JellyMonitor (French et al., 2018) - a self-contained system that is designed to monitor coastal environments and notify

about the presence of several objects and phenomena, jellyfish being one of them. In this Chapter, we outline several issues that are present in the classification part of the project and propose a series of enhancements that are aimed to solve these problems. We empirically prove that our solution is more robust to false detections, generalises better to various types of underwater environments, and does not add much computational overhead.

The major contributions of this Chapter are summarised as follows:

1. Robust detection and classification of underwater objects, in an unmanned real-time manner;
2. Application of generative adversarial networks to sonar imagery;
3. A two-step classification framework that allows for fast, independent training and better results at test time.

3.1 Data Overview

Data collection is an iterative process for the JellyMonitor project. It is performed by placing the imaging sonar underwater, together with a battery and a hard drive that stores captured footage. We use Nvidia Jetson TX2 as the embedded processor which has a built in graphics processing unit that allows to execute deep neural networks offshore quickly. At the initial stages of this research, it was important to configure the range of the acoustic device, collect initial data, perform battery durability tests, and make sure that the system can sustain underwater. This was done in tank and then in a harbour, which resulted in necessary adjustments, as well as many clear images of moon jellyfish (Aurelia Aurita), as they were released manually within the sonar's field of view. Then, to obtain footage in realistic scenarios, where the presence of human actions is minimal, JellyMonitor was deployed on the seabed off the coast. The deployment was successful and brought a large amount of data (Table 3.1, Year 2017). Such large quantity of imagery contains millions of objects; even with automated image analysis, it would require months, if not years, to

Table 3.1: Summary of collected imagery and labelled objects. Please note that an object is always comprised of many frames, up to 300. Bg = Background, Jf = Jellyfish, Ar = Artefacts, Fi = Fish, Sw = Seaweed, Sd = Sediment.

Description	Total Frames	Bg	Jf	Ar	Fi	Sw	Sd
2015 Trial	186,164	56	179	138	0	0	0
2015 Trial	180,277	98	57	49	169	38	0
2016 Trial	598,926	77	8	109	67	88	13
2017 Deployment	14,119,741	281	70	703	565	372	175
2018 Deployment	1,903,140	1675	44	370	1006	48	1670
2019 Deployment	2,343,344	3313	223	486	4583	1010	1278
Total	19,331,592	5500	581	1855	6390	1556	3136

spot and label the main targets of interest, such as jellyfish, seaweed, and fish. Therefore, we changed our data collection strategy for all the next deployments - the system would be turned on only for short periods of time between tides, when tidal flow is at its fastest. Jellyfish are stolid animals, and the chances of spotting them are much higher when marine currents are present. Moreover, it allows for the battery to last longer, which leads to more observations. Such drastic changes in underwater environments over the years resulted in several datasets that are fundamentally different from each other. In a controlled tank environment, higher image quality is expected due to well-regulated conditions, resulting in sharper and more consistent sonar images. From a machine learning standpoint, this data is advantageous, as it enhances the precision and reliability of object detection and classification algorithms. In harbour settings, image quality might vary, influenced by factors like water movement and water quality. Moreover, jellyfish that are manually put next to sonar are not guaranteed to stay within the frame, which makes it more difficult to obtain desired data. The open sea presents its own challenges - reduced visibility, greater water depth, and fluctuating lighting conditions, all of which can pose substantial hurdles for machine learning algorithms. Finally, those three environments present very different class representations and ratios - for example, there were no fish, seaweed, and sediment in tanks, and background was rather clear. The harbour did not experience a lot of water turbulence, and therefore there was no sediment. However, movement patterns of detected phenomena were still different when we compare the imagery to footage captured in the tank. The sea

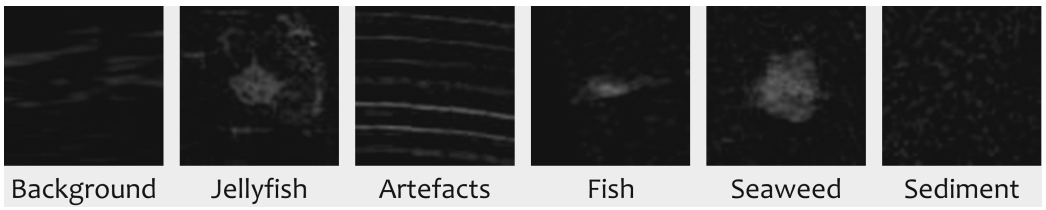


Figure 3.1: Examples of processed sonar images of categories present in the datasets.

comes with a large number of its own properties, such as different seaweed appearances, fish species, sediment, and noise levels, which impacts the way sonars build imagery. Spotted objects are put in six distinct categories, which are described as follows:

- Background - noise that surpassed thresholds during the denoising process and caused a false detection;
- Jellyfish - mostly moon jellyfish, the main target of interest;
- Artefacts - sonar signal reflections from the water surface;
- Fish - a very common object in any waters;
- Seaweed - countless species of marine plants that can also cause damage to the industry;
- Sediment - air bubbles and diffuse clouds of sediment that are caused by water turbulence.

The visual representation of classes is shown in Figure 3.1. Before series of extracted patches are fed into the CNN classifier and predictions are produced, collected data goes through coordinate system conversion, noise reduction, object detection and object tracking steps. All detected phenomena are later manually annotated by a single human observer. Although alone, the human observer possesses an extensive familiarity with sonar data and has been an integral part of the JellyMonitor project since its inception, having examined numerous frames collected in tanks, making him the most qualified individual for labeling tasks in this project. This Chapter focuses solely on improving classification results and, therefore,

does not propose any modifications to the other parts of the system. A more detailed description of the pre-processing pipeline and the labelling application can be found in the previous work (French et al., 2018). We stress that when we mention *objects* (also *events*), we talk about a set of *frames* (also *patches*) that are extracted by the tracking algorithm and contain a single entity.

3.2 Method

This Section addresses poor JellyMonitor generalisation and proposes a number of improvements that aim to increase frame, event, and jellyfish accuracies, as well as lower the amount of jellyfish false positives, in that particular order.

3.2.1 Baseline system

Earlier experiments did not consider leave-one-out cross-validation on different folds of acquired datasets; hence, the reported results did not demonstrate the model's ability to generalise effectively. This was because both the training and test sets were drawn from the same underwater environments. In other words, all imagery was combined and then split into 75%/12.5%/12.5% chunks for training, validation, and testing, respectively. In real-life scenarios, test data typically originates from slightly different environments, characterised by varying levels of noise and artefacts. To address this limitation, we have shifted our evaluation strategy. Instead of randomly drawing splits from all available footage, we now employ separately acquired datasets for training and testing. These separately acquired datasets differ from the original dataset in several crucial ways. Firstly, they represent distinct underwater environments, each with its unique set of conditions, such as water clarity, lighting, and the presence of various marine species. Secondly, these datasets may exhibit varying levels of noise, distortion, and artefacts that are specific to their respective acquisition scenarios. This change in approach allows us to assess the model's performance under more realistic conditions, where the test data is inherently dissimilar to the training

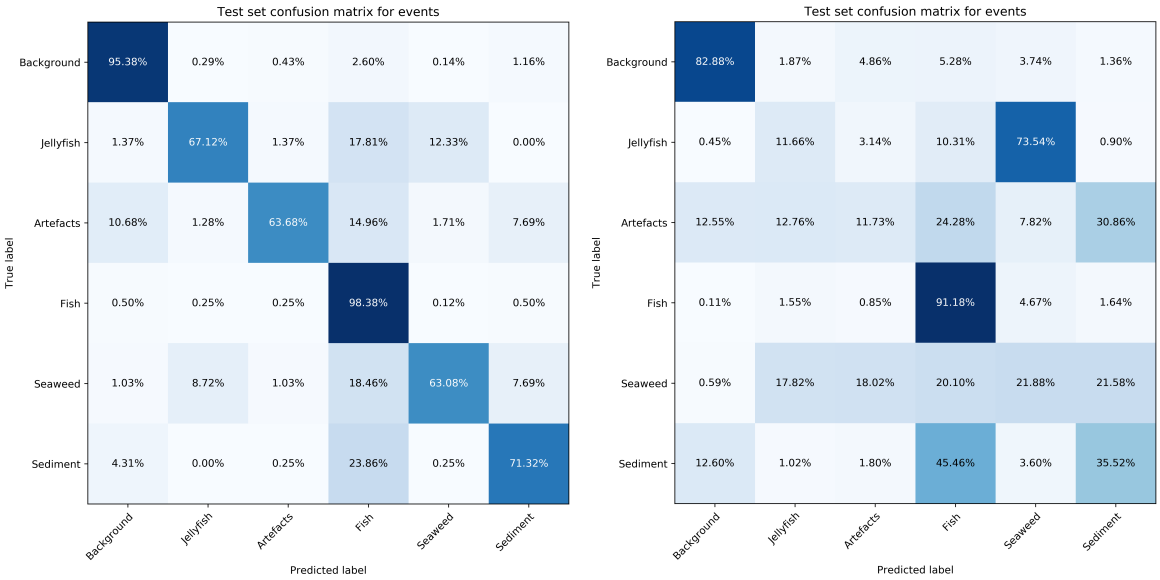


Figure 3.2: JellyMonitor evaluation by drawing random splits from all the available imagery (left) and using leave-one-out cross validation (right, training on Years 2015-2018, testing on Year 2019). The latter approach shows how the classifier fails to generalise on most classes.

data. Figure 3.2 shows the importance of testing on a dataset that is not present in the training split. We note arguably the most challenging problem of the JellyMonitor project - confusion between jellyfish and seaweed. The classes share similar acoustic properties, such as low density and irregular shapes, often making them appear alike. This is further amplified by the fact that sonar imagery often lacks fine details and is grayscale, stripping jellyfish and seaweed of important visual features. Moreover, these objects exhibit similar movement patterns in water currents. We address this issue throughout the Chapter and propose an improvement that is specifically aimed at alleviating this confusion in Section 3.2.3. Throughout this Section, we present several modifications that are aimed to improve classification results. To show their effectiveness, we report performance on the most recent dataset, Year 2019. It has the highest number of labelled objects, contains 38% of all spotted jellyfish, and was collected in real life conditions (Table 3.1). All reported results are calculated as an average over 10 randomly initialised runs. Table 3.2 shows results achieved by various classification designs proposed in this Section.

Table 3.2: Results for various proposed classification frameworks. For frames, events and jellyfish, we report mean accuracy (higher is better). For jellyfish false positives (FP), we report the proportion of objects that are mislcassified as jellyfish (lower is better). All the upgrades are cumulative, meaning that weighted loss, for example, also has the benefits of generated data and the event classifier.

Method	Frame	Event	Jellyfish	Jellyfish FP
Baseline system (French et al., 2018)	74.21%	69.80%	11.52%	2.26%
+ Generated data (Section 3.2.2)	**75.05%**	68.75%	10.85%	1.48%
+ Event classifier (Section 3.2.3)	75.05%	**69.88%**	18.07%	1.74%
+ Weighted loss (Section 3.2.3)	75.05%	69.87%	**38.51%**	2.51%
+ Confidence threshold (Section 3.2.4)	75.05%	69.87%	30.16%	**0.91%**

3.2.2 Lack of data

Footage collection and annotation is an expensive and tedious process. Moreover, the objects of interest, such as jellyfish and seaweed, are not guaranteed to be present, which further amplifies the difficulty of acquiring desired imagery and results in an imbalanced dataset. Since deep neural network classifiers benefit from large amounts of samples, as that leads to better generalisation, data generation becomes one of the solutions. We implement StyleGAN (Karras et al., 2019) to supply synthetic samples in order to expand the training dataset. The StyleGAN framework is composed of two models: the generator and the discriminator, which are trained together. The purpose of the discriminator is to assign correct labels to both real and generated samples, while the generator is trained to minimise correct predictions by the discriminator. The networks take turns in a minimax game, where improving the performance of one network negatively impacts the performance of the other. This leads to the non-saturating loss:

$$\begin{aligned} L_D &= log(D(x)) + log(1 - D(G(z))), \\ L_G &= -log(D(G(z))), \end{aligned} \tag{3.1}$$

where D is discriminator, G is generator, and z is random noise. Generated data $G(z)$ is labelled as 0, and real data x is labelled as 1. Once trained, the discriminator is discarded, and the generator acts as a transform function which turns a fixed-length random vector into

a sample that matches the desired distribution. An important extension to this framework is conditioning (Mirza and Osindero, 2014) - inputs to both models are accompanied with class labels. This way, the networks learn specifics of each category, and later can be used to generate samples of a given class. Both StyleGAN and conditional GAN are covered in Section 2.4.

We leave models' architectures, hyperparameters and the training process unchanged, and train the networks until the discriminator has seen 25 million real images - the number of images recommended to use when training StyleGAN architectures from scratch (Karras et al., 2019). We do not use the style-mixing feature proposed by Karras et al. (2019) to generate samples, though we leave it on during the training to enhance the quality of produced images. Both real and generated visual examples for each class can be found in Figure 3.4. After being trained on Years 2015-2018, the generator is used to produce 600 000 samples in total, 100 000 per category. To inspect the benefits of training on fake data in detail and find the most optimal strategy, it was decided to try 3 different enhancement strategies, shown in Figure 3.3. Adding samples proportionally to the real ones (Figure 3.3a) retains the same ratios between classes, which has very little chance of introducing data bias and, hence, increasing the number of false positives and false negatives. The second approach (Figure 3.3b) is designed to fill the gaps and allow classes, such as Jellyfish and Seaweed, to have more impact on the loss which might force the CNN to learn visual features related to those categories better. Finally, adding the same number of images to each class (Figure 3.3c) blends the preceding strategies together - it has little impact on category balance yet allows to add reasonable portions of samples to each.

We experimented with adding 10%, 20% and 50% of fake samples with respect to the real data. For each setup, we report mean frame and event accuracy to show overall performance, as well as the sum of standard deviations over true positives to capture variance in

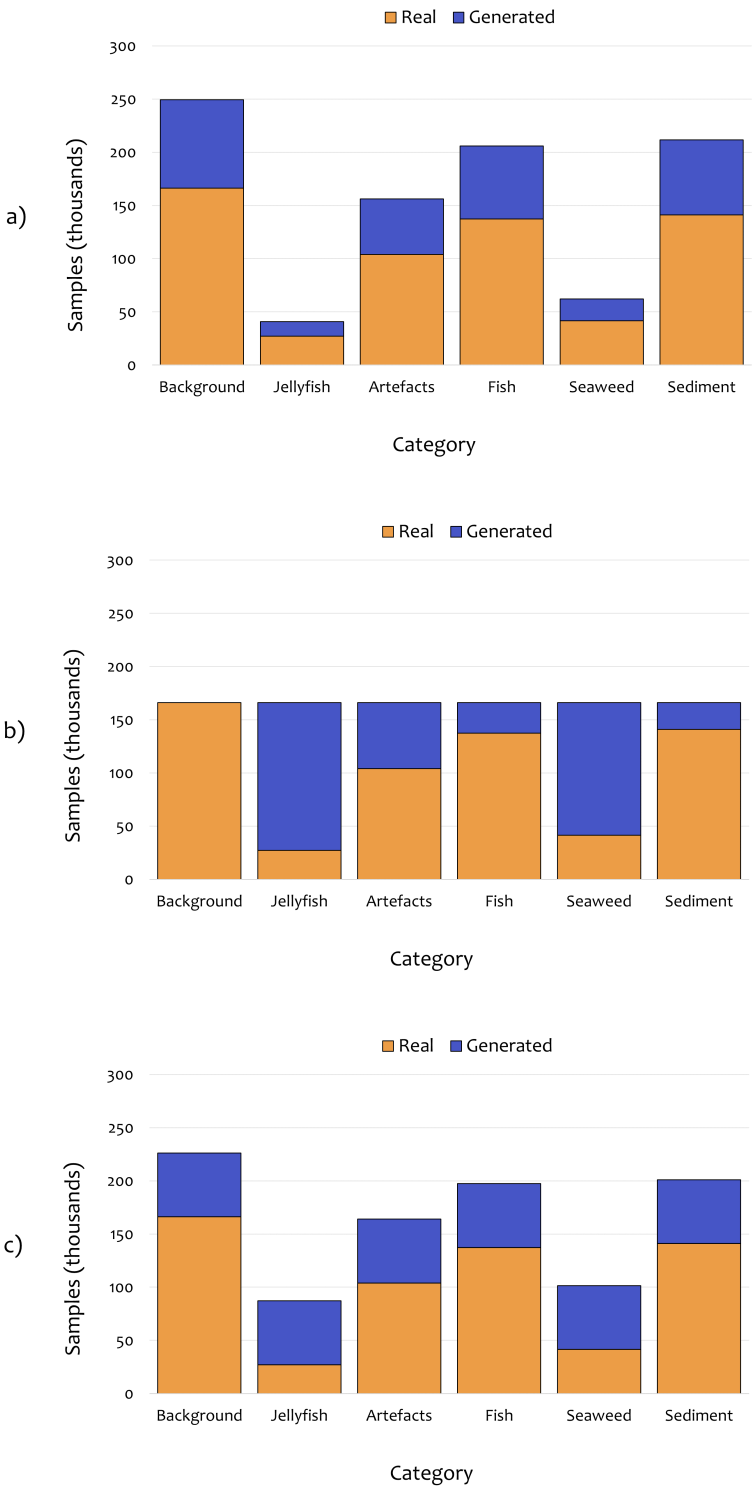


Figure 3.3: The chosen strategies for enhancing the train set with generated data: a) proportional to real data, b) proportional to missing data, c) adding the same number of samples to each class.

Table 3.3: Mean frame and event accuracy and sum of standard deviations for frames and events for various data enhancement setups. The referred strategies are shown in Figure 3.3.

Strategy	Frame	Event	Frame std	Event std
Only real data	74.21%±1.30	**69.80%±1.38**	**1.00**	**1.00**
a) at 10%	74.19%±1.58	68.00%±1.05	1.12	1.13
a) at 20%	73.60%±1.01	67.22%±1.08	1.14	1.16
a) at 50%	74.32%±1.93	67.68%±2.15	1.38	1.51
b) at 10%	74.32%±1.74	68.39%±1.47	1.28	1.36
b) at 20%	72.35%±1.99	66.79%±2.18	1.36	1.38
b) at 50%	71.21%±2.28	66.36%±1.82	1.87	1.89
c) at 10%	**75.05%±1.22**	68.75%±1.08	1.10	1.09
c) at 20%	73.49%±3.16	68.21%±1.95	1.36	1.36
c) at 50%	71.94%±2.53	65.86%±2.22	1.93	1.85

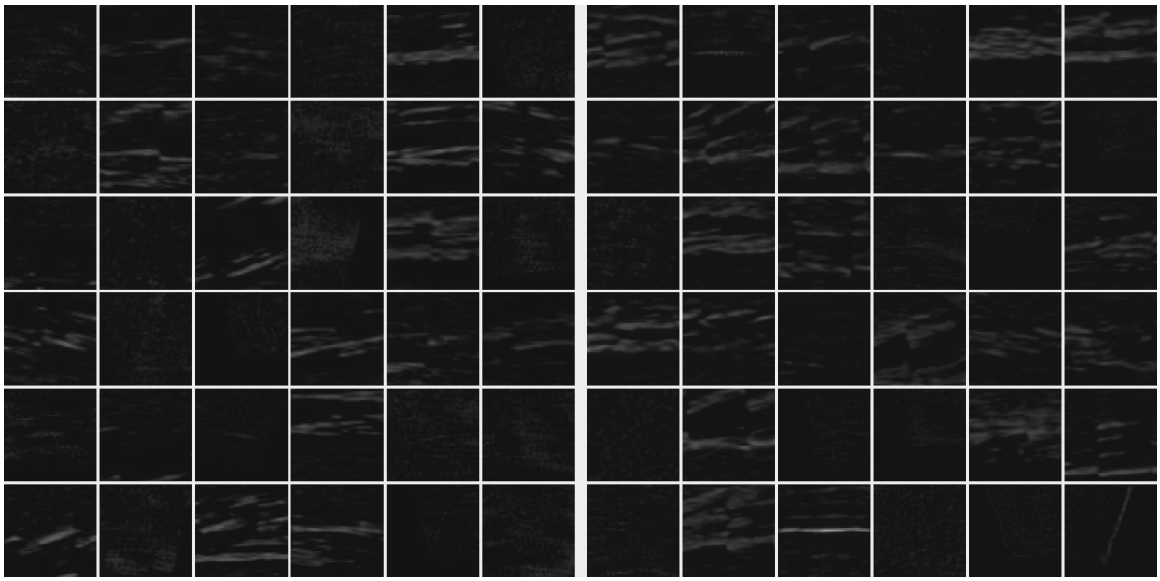
predictions. We calculate the latter as follows:

$$y = \sum_{c=1}^{k} \sqrt{\frac{\sum_{i=1}^{n} (c_{x_i} - \overline{c_x})^2}{n-1}}, \tag{3.2}$$

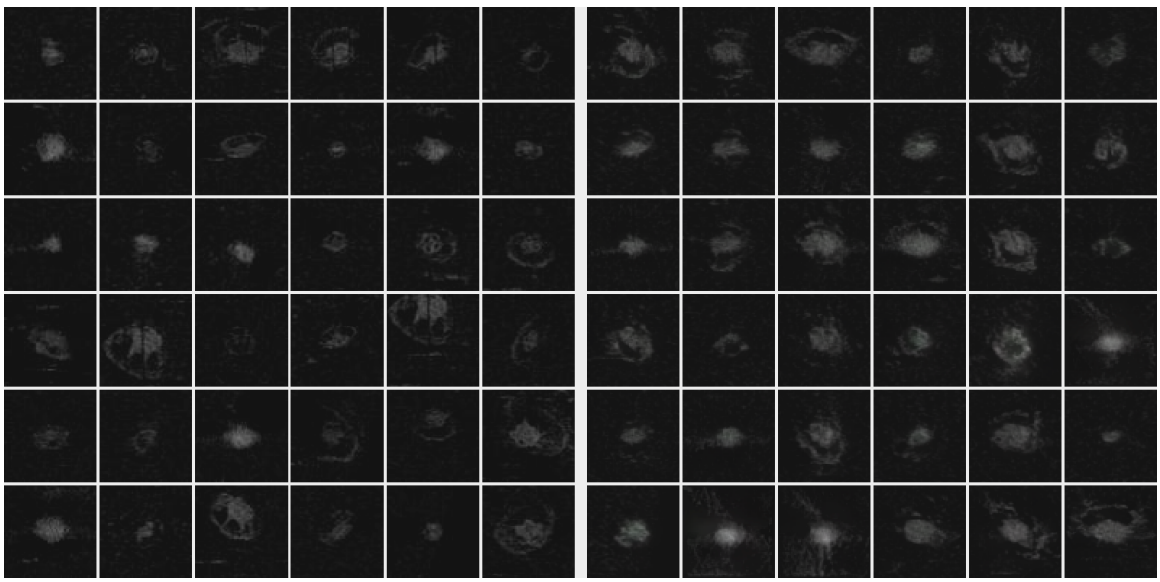
where k is the total amount of classes, n is the total amount of confusion matrices and c_{x_i} is the number of true positives for a particular run. For better readability, we normalise these numbers by dividing them by the value obtained for the first configuration. The results are shown in Table 3.3.

Clearly, adding synthetic data in small portions has the potential of slightly improving the system's performance. However, as the amount of fake images increases, a rapid growth in variance can be observed. We hypothesize that this behaviour could be caused by two factors:

1. The generative model learned the training datasets too well. Therefore, the classifier becomes more biased when given additional imagery that lies within the same distributions. As a result, the model assigns incorrect labels to samples which are specific to that particular test set, e.g.: artefacts;

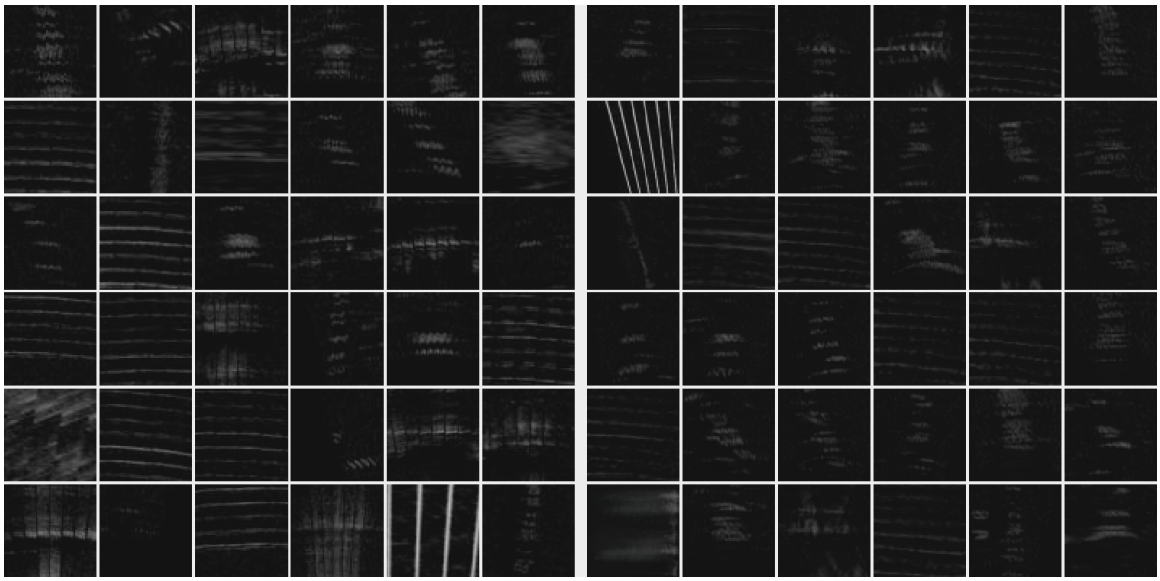


(a) Examples of real (left) and generated (right) Background patches

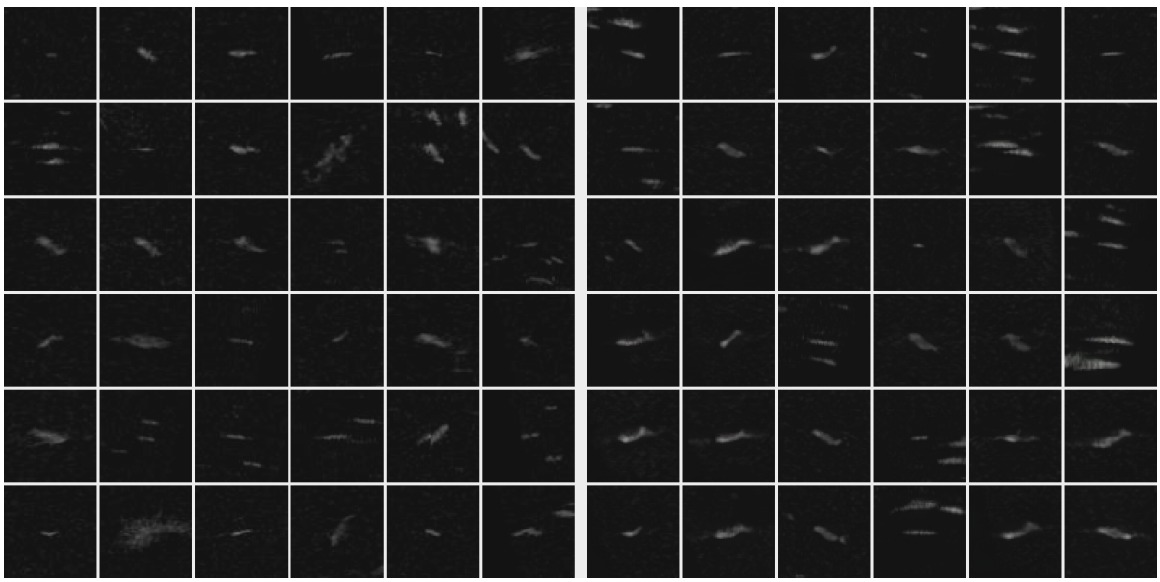


(b) Examples of real (left) and generated (right) Jellyfish patches

Figure 3.4: Examples of real and generated patches for each category.

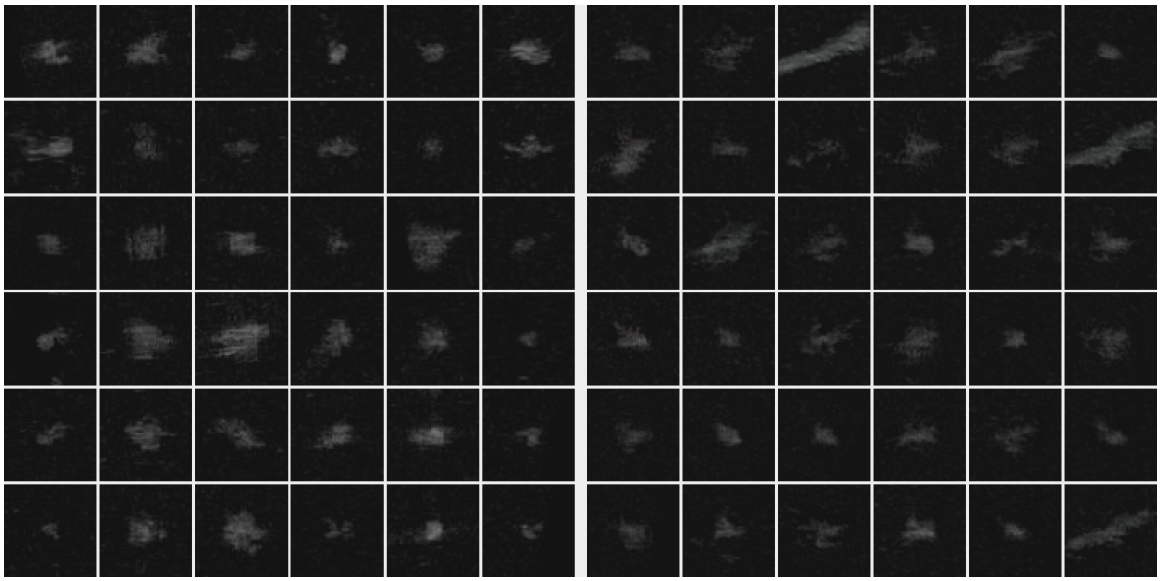


(c) Examples of real (left) and generated (right) Artefacts patches

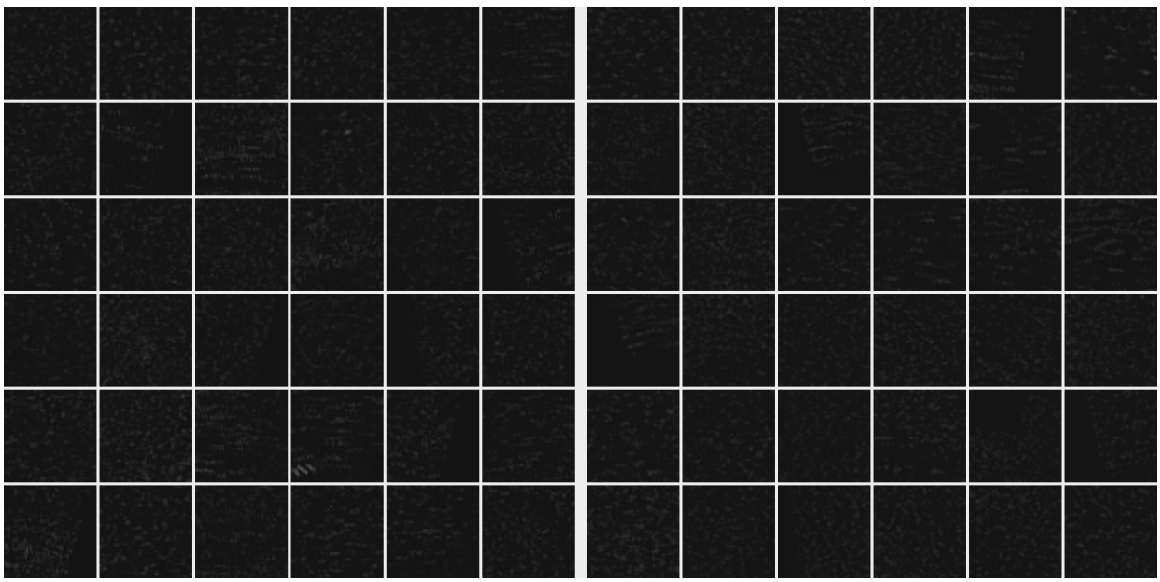


(d) Examples of real (left) and generated (right) Fish patches

Figure 3.4: Examples of real and generated patches for each category (continued).



(e) Examples of real (left) and generated (right) Seaweed patches



(f) Examples of real (left) and generated (right) Sediment patches

Figure 3.4: Examples of real and generated patches for each category (continued).

2. The generative model produced too many noisy images which further complicates the classifier's training process.

We stress that enhancing strategies are as important as the quality of fake data. The more the balance between classes is affected, the more variance in results can be observed, as well as drops in accuracy. Strategy `a)` kept ratios the same, and had the least negative impact in that regard. We chose the setting that has achieved the highest frame accuracy to work with in the subsequent experiments, i.e. strategy `c)` at 10%. Whereas an improvement of 0.84% in accuracy might seem not large enough, we stress that the ceiling is much lower than 100% due to the human error in labels, as well as large amount of noise present in the data (e.g.: background frames are met in sequences, when the object is mostly faded but still being tracked). A paired t-test was conducted to assess the difference in mean accuracy between "Only real data" and "`c)` at 10%." The results indicated a statistically significant difference ($t = 988.68$, $df = 2{,}343{,}343$, $p < 0.05$), with "c) at 10%" showing higher mean accuracy (75.05%) compared to "Only real data" (74.21%) with a sample size of 2,343,344, as shown in Table 3.1. Although event accuracy was not improved, we address this issue and focus on it in the next subsection.

3.2.3 Event information fusion and weighted loss

When the system stops tracking an object, a series of frames is extracted and passed to the classifier CNN. In this framework, each event comprises a varying number of images, typically ranging from 4 to 300, with an average of approximately 76 images per event. The classification process operates on a per-frame basis, where each frame is individually classified. Then, the framework computes the average prediction for all frames associated with the same object, resulting in a single output vector per object that contains class probabilities. In other words, it employs a standard classifier that sequentially classifies frames and then combines the predictions for frames related to a single object, as shown in Figure 3.6a). Although this approach yields fairly good results, it relies on the idea that the majority of frames are classified correctly. However, often that is not the case - some objects

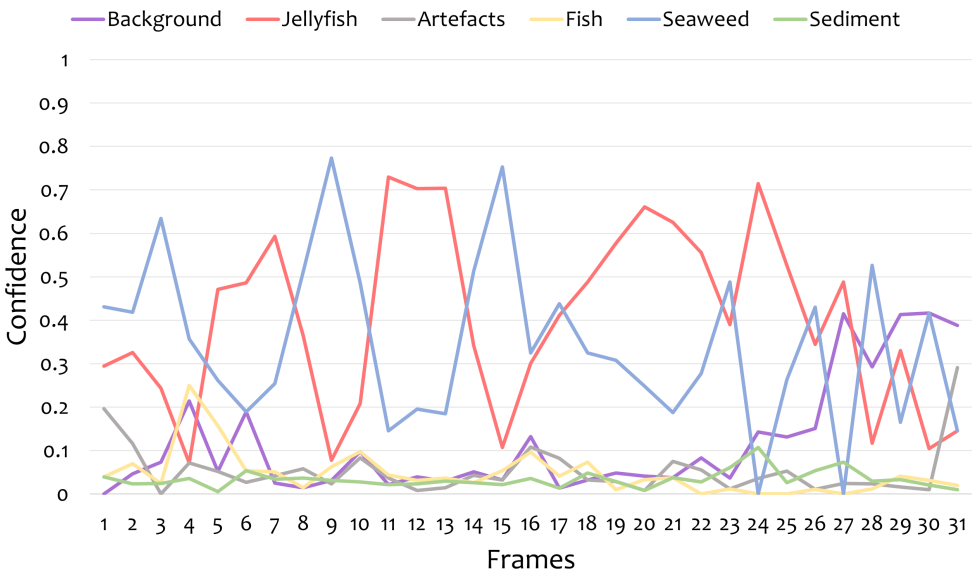


Figure 3.5: Confidence scores over a number of frames for a jellyfish. The pulsating motion causes a lot of variance in predictions - when the jellyfish contracts, it looks very similar to seaweed. Moreover, the first and last frames might often introduce additional noise, in cases when an object enters or exits the sonar's field of view, resulting in the object being visible only partially.

might drastically change in shape and size. The problem is further amplified by changes in noise and artefact appearance. Deep neural networks are known to be very sensitive to even the slightest changes in data, and this results in incorrect predictions, which creates a gap between frame and event accuracies, as seen in Table 3.2, Baseline and Generated data methods. Figure 3.5 shows examples of rapid fluctuations in per-frame predictions. To combat this, one would have to come up with a method that considers objects' metadata instead of just taking the average. We experimented with changing the input to several images, adding mean and standard deviation over frames, and found that a network which convolves over confidence scores for series of frames in a sliding window manner works the best. The latter idea is shown in Figure 3.6.

The motivation behind the proposed framework came from manual analysis of confidence patterns, where we noticed that common features are often shared within one category, e.g.: small variance in shape for seaweed which leads to rather constant predictions over frames,

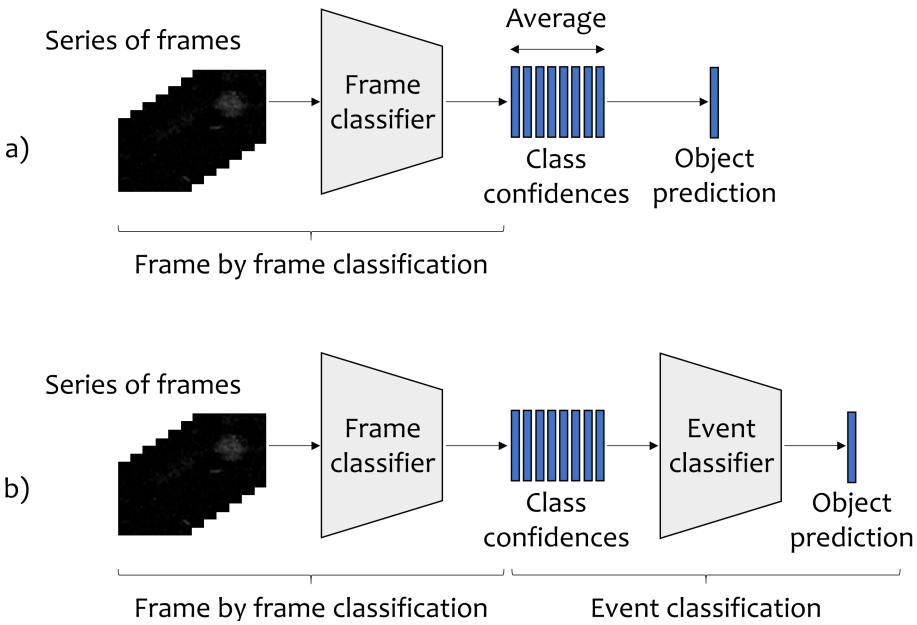


Figure 3.6: Visualization of two object classification frameworks: a) taking average over all individual frame predictions that belong to the same event to produce the final result (French et al., 2018); b) using an additional neural network to learn from confidence patterns, proposed in this Chapter.

pulsating motion for jellyfish that causes frequent changes, demonstrated in Figure 3.5. If these patterns are consistently met across the datasets, the system can learn from them and reduce the confusion between certain classes as a result. Another benefit of the event classifier network, when compared to other techniques, is the small computational overhead, both in memory and time, which we analyse in detail at the end of this subsection. The models' architecture is rather lightweight, and is comprised of only two convolutional and one classifier layers, with adaptive average pooling connecting the blocks, as seen in Table 3.4. The architecture was found heuristically - we needed a compact network that would work in a sliding window manner. Therefore, the amount of potential layers was very limited, leaving only the kernel size to experiment with. We found that 6 x 4, followed by 6 x 6, produced the desired result. Table 3.5 compares the two classification frameworks presented in Figure 3.6. To keep the comparison fair, the same pre-trained frame classifiers were used when swapping techniques.

Table 3.4: Event classifier architecture.

Description	Activation	Output shape	Parameters
Input confidences	-	1 x 6 x n	-
Conv 6 x 4	LReLU	128 x 6 x n	3 200
Conv 6 x 6	LReLU	128 x 6 x n	589 952
Adaptive Avg Pool	-	128 x 6 x 1	-
FC	Softmax	1 x 6 x 1	4 614
Total			597 766

Table 3.5: Event accuracy for 10 randomly initialised runs using averaging and event classifier.

Run #	Averaging	Event classifier
1	68.25%	69.61%
2	67.43%	69.85%
3	68.88%	69.29%
4	68.47%	69.44%
5	66.98%	68.72%
6	68.72%	69.10%
7	68.99%	69.34%
8	69.18%	71.50%
9	70.63%	70.64%
10	69.96%	71.36%

The suggested change outperformed averaging on every run by a margin of up to 2.42%. However, the main purpose of JellyMonitor is to reliably recognize jellyfish without triggering any false alarms which can be caused by many false positives. Therefore, assessing the system's performance solely on frame and event accuracy is not the best approach when it comes to our application. Accuracy is mostly affected by categories with large amounts of samples, such as Background, Fish, and Sediment. Whereas it is important to keep this number high for monitoring purposes, a change that can bring the number of Jellyfish true positives up at the cost of some percentage of Sediment being classified as Background, for example, is considered a positive trade-off. That leads to the idea of weighted loss - scaling the error up for some particular classes forces the model to assign correct labels more often when given samples from those categories. When reporting results, we also include Jellyfish accuracy and the number of false positives, which induce the risk of triggering alarms. The

main confusion is between Jellyfish and Seaweed, as they happen to look similar on many frames. Moreover, these two are the least popular classes in the training dataset - only 546 Seaweed and 358 Jellyfish (Table 3.1), mostly captured during trials, which are different to real underwater conditions. Hence, we experimented with scaling up the weights for them, such that traditional cross-entropy loss:

$$l = -\sum_{i=1}^{k} y_i * log(p_i), \tag{3.3}$$

where k is the total amount of classes, y is the true class labels in the form of a one-hot vector, and p is the predicted probability for the true class labels, becomes:

$$l = -\sum_{i=1}^{k} w_i * y_i * log(p_i), \tag{3.4}$$

where k is the total amount of classes, w is the class weight, y is the true class labels in the form of a one-hot vector, and p is the predicted probability for the true class labels. We use the weighted loss to generate results shown in Table 3.6. It was impossible to make use of synthetic data to train the event classifier, as the generative network does not produce videos, only individual frames. Although we considered using weighted loss for the frame classifier too, the attempts did not lead to a desired accuracy increase, mostly due to the fact that the classifier is very sensitive to data imbalance.

Clearly, forcing the network to perform better on Jellyfish and Seaweed does not affect the overall event accuracy too much, as seen in Table 3.6. There is an evident relationship between Jellyfish true and false positives - bringing Jellyfish accuracy up introduces more false detections, and vice versa. However, it is important that JellyMonitor is able to spot a bloom, hence, we choose the setting that has achieved the highest Jellyfish accuracy for our further experiments. A paired t-test revealed a significant difference in accuracies between the initial and chosen loss setups. Initial results (all classes set at 1.0): mean accuracy of 18.07% with a standard deviation of 7.44. The chosen loss setup (2.0 for Jellyfish, the rest

Table 3.6: Mean event and jellyfish accuracy and jellyfish false positives for different weighted loss setups. X and Y are the weights for Jellyfish and Seaweed, respectively. All the other classes were set at 1.0. False positives range from 0 (best case) to 16250 (worst case).

X, Y	Event	Jellyfish	Jellyfish FP
1.0, 1.0	69.88%±0.95	18.07%±7.44	282.1±143.7
1.5, 1.0	**69.96%±1.10**	28.48%±6.64	383.0±168.7
1.5, 1.5	69.85%±1.05	20.36%±6.29	315.7±147.7
1.5, 2.0	69.71%±1.09	16.10%±6.23	278.8±140.2
2.0, 1.0	69.87%±0.95	**38.51%±9.85**	408.6±168.2
2.0, 1.5	69.81%±1.05	29.69%±7.15	391.5±168.2
2.0, 2.0	69.58%±1.01	21.75%±7.18	380.1±237.0
2.0, 3.0	69.58%±0.97	15.56%±6.80	**268.6±128.1**
3.0, 2.0	69.58%±0.99	32.60%±7.49	423.3±175.0
3.0, 3.0	69.53%±0.96	22.38%±8.37	382.1±210.9

at 1.0): mean accuracy of 38.51% with a standard deviation of 9.85. The test, conducted with 223 samples, as shown in Table 3.1, indicated that our choice of weighted loss exhibited a significantly higher mean accuracy compared to the initial setup ($t = 6.12$, $df = 222$, $p < 0.05$). We focus on eliminating false positives in the next subsection.

It is crucial that the proposed technique does not introduce a lot of computational overhead - predictions should be performed in real time, and any major slow downs that could potentially lead to accumulation of data to be processed must be avoided. The event classifier performs well in that regard, slowing the system down by only up to 25% for frame lengths of 150 and less, as shown in Figure 3.7. Given that now footage capture happens only when the tidal flow is at its fastest, and objects move quickly, we do not expect many to surpass this threshold. As for GPU memory, it introduces additional 0.6 GB during test time with the current settings. When added to the initial 0.8 GB that come with the frame classifier, it is still way below 8 GB - the capacity offered by Nvidia Jetson TX2.

The proposed framework not only was able to provide a more structured approach to object classification, but also has given an opportunity to set priorities for classes with the help of

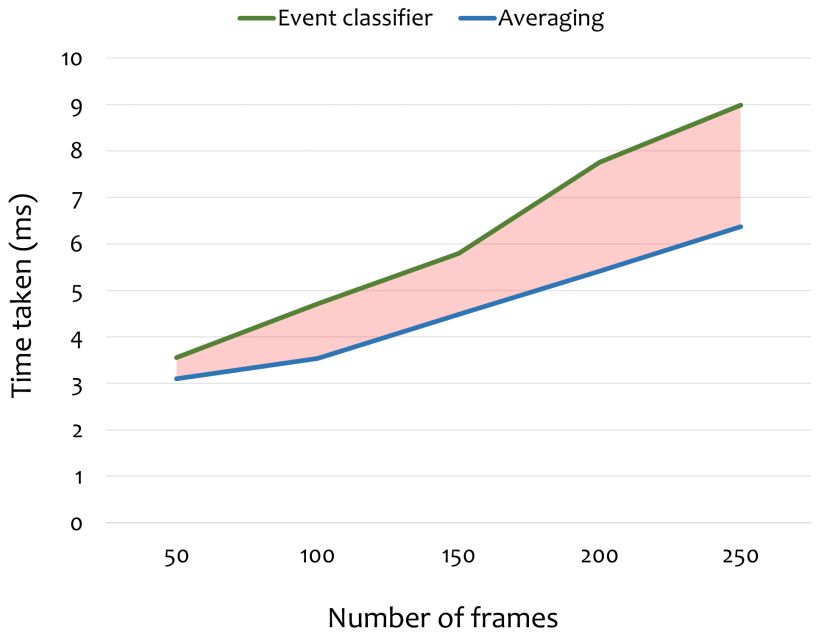


Figure 3.7: Average time taken to classify events with different frame lengths before and after adding the event classifier. Pink area shows the additional time overhead.

weighted loss. That resulted in a significant boost of correctly classified jellyfish, however, introduced more false positives. Data collection strategy plays an important role - if before Year 2018 underwater phenomena moved freely most of the time, now they are almost always susceptible to the current, which affects the way objects change in shape over frames. Therefore, we believe that the event classifier will have much more positive impact during the next deployment, once Year 2019 is included in the training split.

3.2.4 Confidence threshold

Objects that are assigned predictions with low confidence lie between major clusters on the classifier's decision surface. This is often an indication that samples are given wrong labels, and the probability of it being true increases as the amount of categories gets larger. It is important that JellyMonitor does not report false alarms - doing so would result in major losses for the industry. Therefore, we employ confidence thresholding, which allows to keep the amount of false positives low, while maintaining satisfactory true positive rate. Jellyfish is the only concern in this case, hence, we do not use thresholding for other classes.

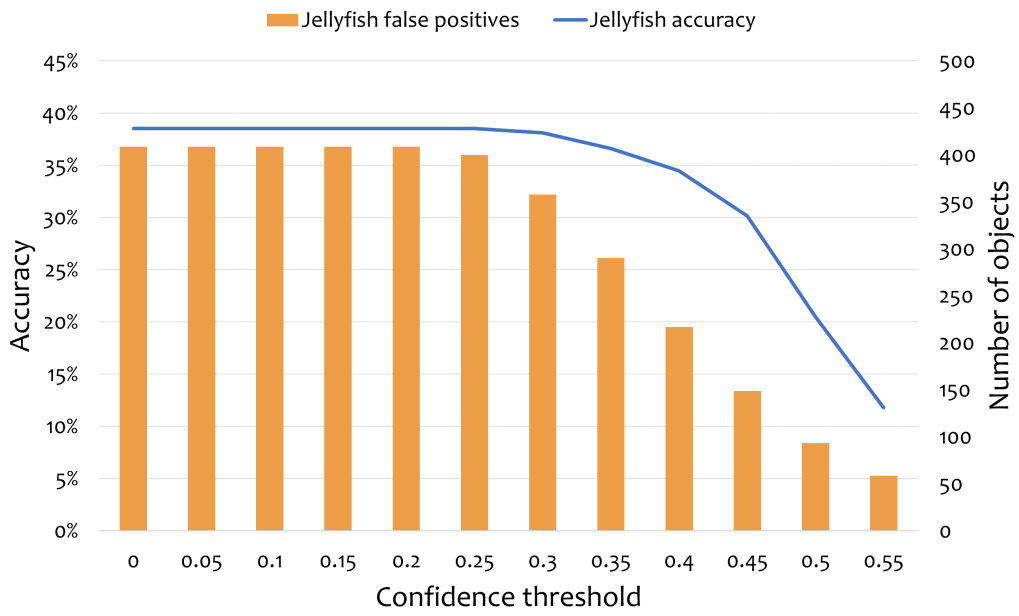


Figure 3.8: Average jellyfish accuracy and false positives for a series of threshold values.

The expected tradeoff can be observed between true and false positives in Figure 3.8. To spot an abundance of jellyfish, it is enough to correctly classify a quarter - even with such a low accuracy, reported numbers will be high enough to indicate a bloom. Therefore, the chosen threshold value is 0.45, offering around 30%, to accommodate for the possible error.

3.2.5 Results

The networks were trained using data from Years 2015-2018 and tested on Year 2019 (Table 3.1). The training set was split into 80%/20% for training and validation, respectively. Both models used learning rate of $1 * 10^{-3}$ with the Adam optimisation algorithm (Kingma and Ba, 2014), cross-entropy loss and early stopping (Prechelt, 1998) with patience of 7. We kept data augmentation for patches the same as in the previous work (French et al., 2018). No data augmentation was useed to train the even classifier.

The final confusion matrix is shown in Figure 3.9. These are the results we would expect the system to achieve in the next deployment, if it was to happen. Events such as Background,

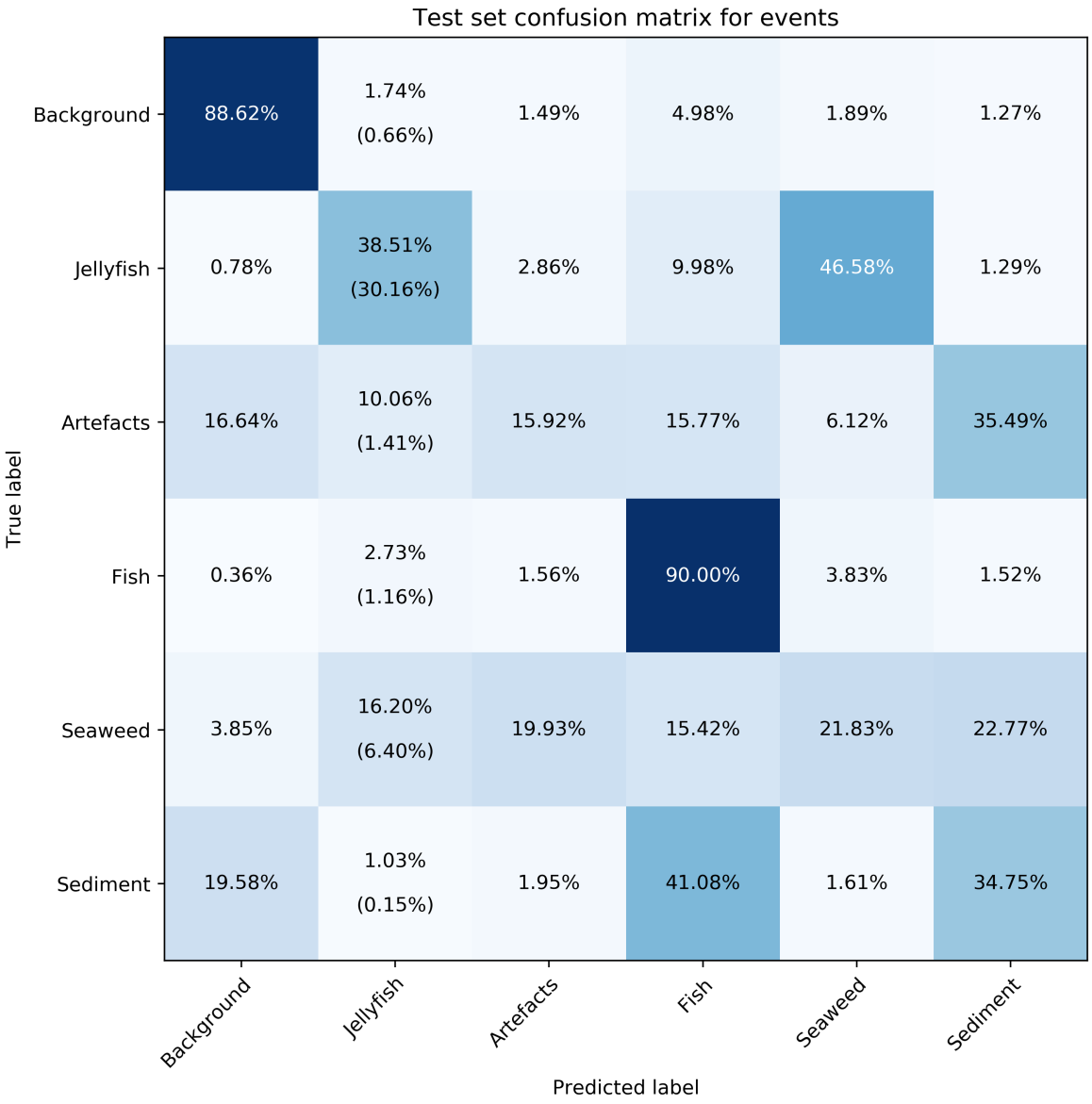


Figure 3.9: Final confusion matrix (average of 10 runs). Values in parentheses represent objects that surpass the threshold and are reported by the system to onshore.

Artefacts, Fish and Sediment would be sometimes misclassified as Jellyfish, however, even when combined together, these classes would not be mistaken as a bloom, due to extremely low error rate. Although Seaweed has a small chance of triggering an alarm, it would need to come in such an abundance that would cause the same negative effects on the industry as gelatinous plankton does. We would like to stress that the accuracy ceiling is much lower than 100% due to the human error that was inevitably introduced during the labelling process, as well as large amount of noise present in the data (e.g.: background frames are met in sequences, when the object is mostly faded but still being tracked, as seen in Figure 3.10).

3.3 Discussion

In this Chapter, we have presented a series of steps aimed at improving the performance of JellyMonitor. We first focused on increasing the frame accuracy by generating additional training data. Then, the event accuracy was also improved with the help of the event classifier network. By employing weighted loss, we significantly enhanced the system's ability to recognise jellyfish. Finally, we reduced the amount of jellyfish false positives by utilising confidence threshold. All together, the proposed enhancements increased the jellyfish detection accuracy from 11.52% to 30.16%, and reduced the amount of false positives by more than 60%, when compared to the baseline (French et al., 2018). The system is now capable of spotting gelatinous zooplankton blooms with high confidence and can lower the damage and costs that jellyfish cause to a number of marine-related industries. Clearly, the system could be adapted to recognise other phenomena, and is not limited to the 6 mentioned categories, therefore, might be applicable to other underwater ecosystems and use cases.

The next stages of the project would include further data collection and annotation, as well as deployments in coastal environments. The plan was to continue employing deep learning methods, in particular, video data generation (Tulyakov et al., 2018) to grow the dataset,

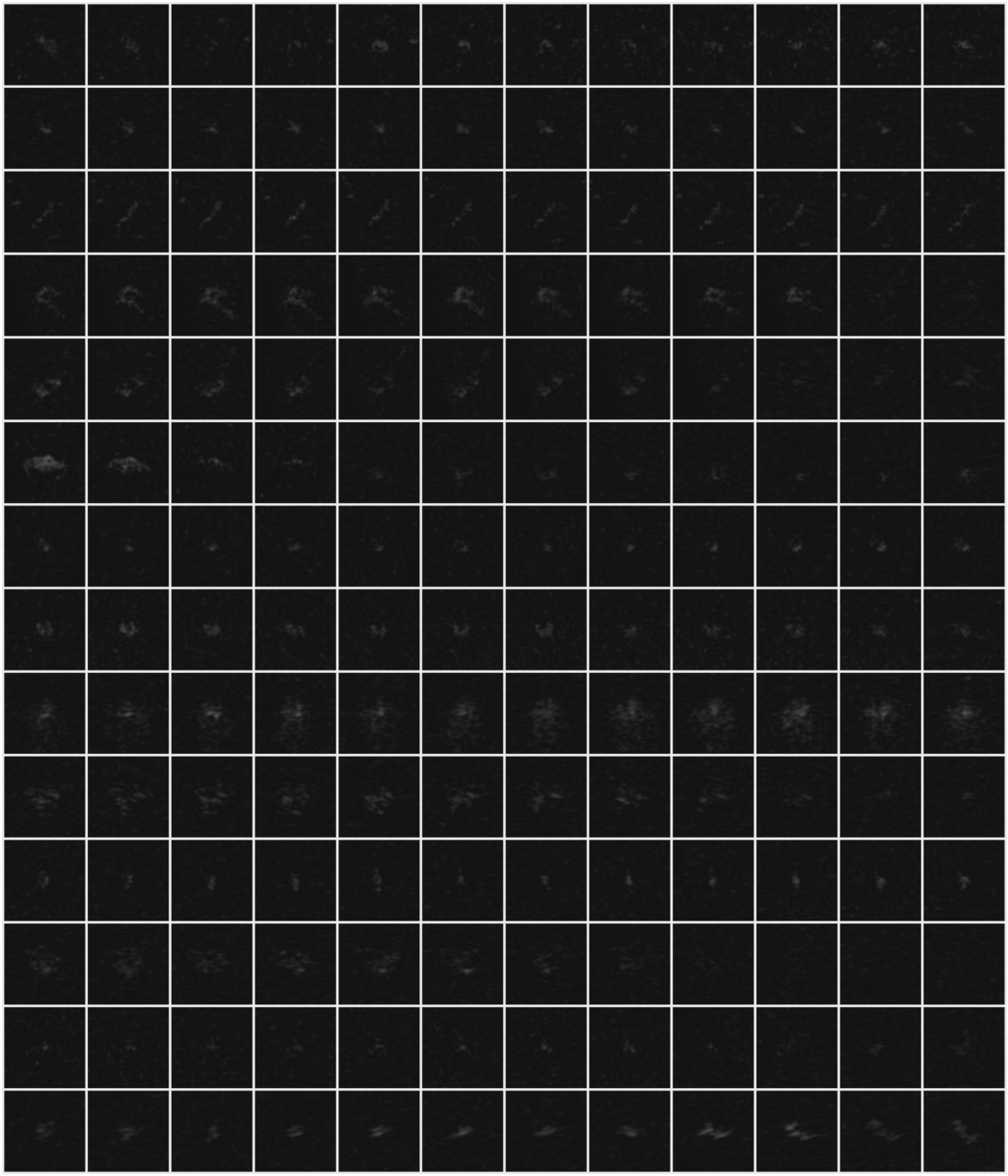


Figure 3.10: Excerpts from sequences of misclassified jellyfish. Each row represents a different object. At the end of most sequences, the object is usually faded but still being tracked.

transfer learning through domain adaptation to enhance the classifier (Ganin and Lempitsky, 2015), and replacing object detection with more sophisticated techniques (Redmon et al., 2016). The use of event metadata, such as motion paths, the speed of objects, the direction of the current, and the distance between the sonar and spotted phenomena could provide additional feature dimensions for event information fusion, and has great potential of improving results.

However, as the project was ceased, we made the decision to continue the research on publicly available datasets. In the next Chapter, we will combine the idea of artificially expanding the dataset with video domain adaptation.

4 Virtual Adversarial Training in Feature Space for Unsupervised Video Domain Adaptation

As discussed in Section 2.6, Virtual Adversarial Training (VAT) is a regularization method initially designed for supervised and semi-supervised learning (Miyato et al., 2018). VAT introduces a concept known as the locally-Lipschitz constraint to the training data, achieved through local perturbations. One of its key ideas is based on the principle that a small change to the input data should not lead to drastic shifts in the model's predictions, aiming for prediction stability. VAT identifies specific directions in the data where these small changes have the most significant impact on the model's predictions. It then introduces minor 'noise' in these directions, a process that mimics real-world data variability and uncertainties. The model is subsequently penalized if it reacts too strongly to these perturbations, fostering a more adaptive and robust behavior.

This approach results in a model that maintains stability, making it a valuable tool in scenarios where labeled data is limited, such as in semi-supervised and unsupervised tasks. VAT has also been successfully applied to image-based unsupervised domain adaptation (Shu et al., 2018; Mao et al., 2019), achieving state-of-the-art results. However, previous methods applied VAT at the pixel level, whereas our exploration focuses on applying it at the feature level. This approach allows for faster computations and doesn't interfere with the feature extraction part of a system, which is often a separate component in video classification models (Chen et al., 2019a; Piergiovanni et al., 2017; Mao et al., 2018).

In this chapter, we delve into the advantages of employing Virtual Adversarial Training in the feature space and apply it to the relatively unexplored domain of unsupervised video domain adaptation. We also address the challenges posed by entropy minimization and decision-boundary iterative refinement training with a teacher (Shu et al., 2018) in domain adaptation, proposing more straightforward alternatives that yield similar results. By incorporating these techniques into the TA^{3}N model (Chen et al., 2019a), we either match or surpass prior results in various unsupervised video domain adaptation tasks.

It's worth noting that contemporary video unsupervised domain adaptation (UDA) algorithms achieve remarkable results in several publicly available adaptation scenarios. Hence, the next logical step in the field is to introduce more complex problems that demand innovative solutions. To this end, we propose a substantial video domain adaptation dataset by merging two extensive datasets - SVW (Safdarnejad et al., 2015) and UCF (Soomro et al., 2012). This new dataset is not only larger in terms of sample size and average video length but also presents additional challenges, including orientation and intra-class variations, resolution disparities, and more pronounced domain differences in content and capturing conditions. An accuracy gap analysis demonstrates that both SVW→UCF and UCF→SVW adaptation paths are empirically more demanding than existing scenarios. Finally, we discuss the characteristics that pose the greatest challenges to modern video domain adaptation methods.

4.1 Method

We begin by briefly describing the TA^{3}N model and losses used for training the network. First, raw videos V are passed through the ResNet-101 model (He et al., 2016) G_r, that was initially pre-trained on ImageNet (Russakovsky et al., 2015), to produce general-purpose features $\hat{X}$, for each frame. The TA^{3}N model uses the extracted feature vectors as the input to perform domain adaptation, as opposed to raw videos and/or frames (Figure 4.1). Then, the vectors are converted to task-driven features $\hat{t}$, by the Spatial module G_{sf}, where the

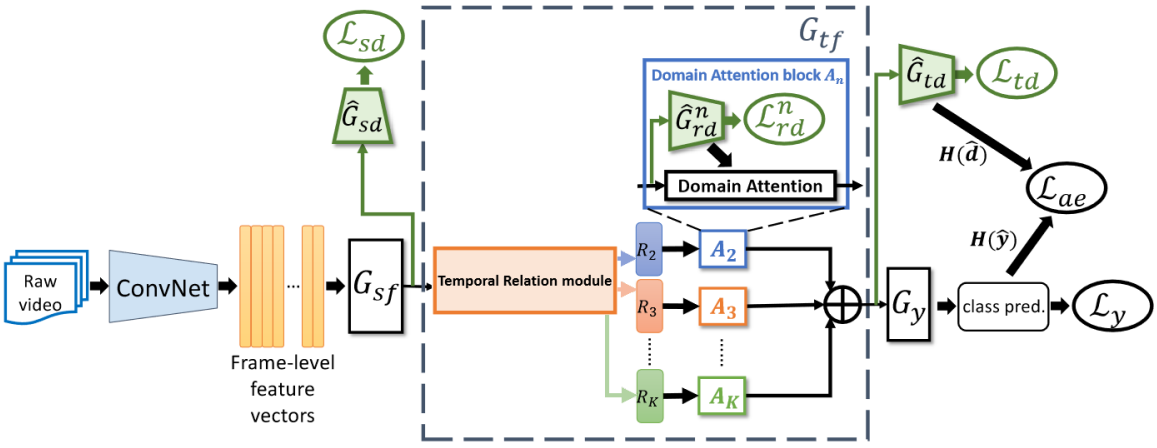


Figure 4.1: Simplified TA^{3}N architecture (Chen et al., 2019a).

task is video classification. Next, the Temporal module G_{tf}, encodes the relation between frames into single vectors, by passing time-ordered sets of 5 frame representations through a multilayer perceptron, and sums them together, resulting in one feature vector $\hat{d}$, for each video. Lastly, the video features go through a fully-connected layer G_y, which produces the final predictions $\hat{y}$. The system makes sure that source and target domains are aligned at all stages by having domain Discriminators included in Spatial, Relation and Temporal modules:

$$L_d = -\frac{1}{N_{S\cup T}} \sum_{i=1}^{N_{S\cup T}} (\lambda^s L^i_{sd} + \lambda^r L^i_{rd} + \lambda^t L^i_{td}), \tag{4.1}$$

where $N_{S\cup T}$ is the number of training samples in both source (S) and target (T) domains, L^i_* is calculated via the cross entropy loss function which is applied to Discriminators' predictions and domain labels, and λ parameters control the weighting of each module. The authors also minimize the entropy for samples that have low domain discrepancy, via the attentive entropy loss:

$$L_{ae} = \frac{1}{N_{S\cup T}} \sum_{i=1}^{N_{S\cup T}} \lambda^{ae}(1 + H(\hat{d}_i)) \cdot H(\hat{y}_i), \tag{4.2}$$

with H being the entropy function:

$$H(p) = -\sum_{i=1}^{n} p_i \cdot log(p_i). \tag{4.3}$$

Lastly, the traditional classification loss L_{cls}, is utilized to minimise the source domain error via cross entropy loss on supervised samples from the source domain:

$$L_{cls} = \frac{1}{N_S} \sum_{i=1}^{N_S} y_i \cdot log(\hat{y}_i). \tag{4.4}$$

The only difference here is that the cross entropy function uses available ground truth labels, y. Putting everything together results in the overall loss:

$$L_{TA^3N} = L_d + L_{ae} + L_{cls}. \tag{4.5}$$

Please note that we omit many details to focus only on the relevant parts and save space. A simplified version of the architecture is shown in Figure 4.1. For the full description of the model, we refer the reader to the TA^{3}N paper (Chen et al., 2019a).

4.1.1 Virtual adversarial training in feature space

For simplicity, let all the aforementioned modules together with the classification layer be G_t, such that

$$G_t(\hat{X}_i) = G_y(G_{tf}(G_{sf}(\hat{X}_i))). \tag{4.6}$$

Traditionally, VAT would seek to minimize the following objective:

$$L_{vat} = \frac{1}{N_{S \cup T}} \sum_{i=1}^{N_{S \cup T}} \lambda^{vat} LDS(V_i), \tag{4.7}$$

where

$$LDS(V_i) = \max_{\|r\| \leq \epsilon} D_{KL}(G_t(G_r(V_i)) \parallel G_t(G_r(V_i + r))), \tag{4.8}$$

where D_{KL} is the Kullback–Leibler divergence, V is the raw video, and r is adversarial noise, respectively. Following TA^{3}N, we freeze the weights of the ImageNet pre-trained ResNet-101, hence, using the adversarial perturbation in input pixel space r_V will have little benefit in comparison to using the feature space perturbation $r_{\hat{X}}$. Given that adversarial perturbations are computed via backpropagation, $r_{\hat{X}}$ is an intermediate value used to compute r_V. Applying r_V would most likely induce the ResNet model to produce features with the similar feature space perturbation to $r_{\hat{X}}$, which would then be used to train the subsequent layers in the network. Therefore, although adding noise to video frames seems logical, there is little benefit in expending the additional computation required to propagate back $r_{\hat{X}}$ through G_r to obtain r_V, only to pass it forward through the ResNet again to recover $r_{\hat{X}}$.

Perhaps, the biggest disadvantage of applying any kind of augmentation to feature vectors is the fact that unlike imagery, it is impossible to visually evaluate the effect of such operations. This introduces additional challenge when it comes to hyperparameter selection. In case of VAT, the main parameter is ϵ, the norm constraint that determines the scale to which adversarial direction is applied to real samples. Such variables are often found via performing multiple experiments and choosing the one that gives the best result. However, this approach is inaccessible in real-world UDA scenarios, where the domain of interest is not annotated. Moreover, input images usually lay in a predetermined pixel range, whereas feature vector values depend on several factors, such as losses, as well as normalization and regularization terms used within the network. These factors affect model weights through backpropagation and make the feature space more sensitive to perturbations than the pixel space. Miyato et al. (2018) note that for small ϵ, both hyperparameters λ^{vat} and ϵ have virtually the same effect on the strength of regularization, and therefore, tuning just one of them is sufficient. We follow their advice and fix $\epsilon = 1$. Therefore, the local distributional

smoothness becomes:

$$LDS(\hat{X}_i) = \max_{\|r\| \leq 1} D_{KL}(G_t(\hat{X}_i) \parallel G_t(\hat{X}_i + r)). \tag{4.9}$$

Although the λ^{vat} value could be found via extensive grid search, we noted that $\lambda^{vat} = 0.01$ consistently achieved good results, hence, we fixed it too, for simplicity. Therefore, our proposed loss becomes:

$$L = L_{TA^3N} + \frac{1}{N_{S \cup T}} \sum_{i=1}^{N_{S \cup T}} \lambda^{vat} LDS(\hat{X}_i). \tag{4.10}$$

We stress that Virtual Adversarial Training is applied to both source and target training features.

4.1.2 Entropy minimisation

Cluster assumption is a fundamental concept that suggests that data points from the same class or category tend to form clusters and should have similar characteristics or behavior. It is used to identify patterns and structure within data by grouping similar data points together, and often assigning them to the same class. Cluster assumption for Domain Adaptation was extensively discussed in DIRT-T (Shu et al., 2018), where it was enforced via conditional entropy minimisation, a technique aimed at promoting the convergence of data clusters via penalising the model for predictions with low confidence scores. However, for it to be successful, the classifier needs to be locally-Lipschitz, which ensures the model's predictions exhibit smooth transitions in the vicinity of data points. In other words, as we move from one data point to another that is nearby, the model's predictions should change gradually and not exhibit sudden, erratic shifts. This constraint was then successfully accomplished by employing VAT. We hypothesise that similar behaviour can be achieved by using VAT alone, as training the network to be locally consistent with its predictions naturally leads to cluster expansion, which indirectly makes the classifier confident about data that lay in the middle of those virtually created clusters, as demonstrated in Figure 4.2. It

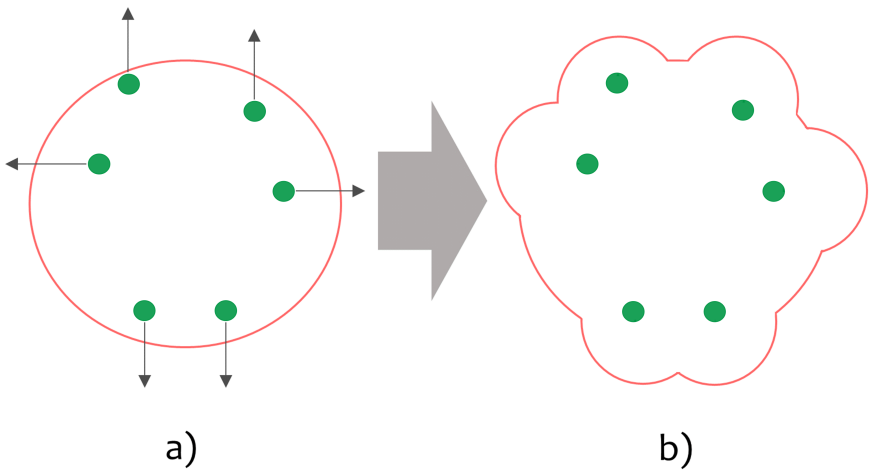


Figure 4.2: VAT cluster expansion. Green - input data points, red - classifier decision boundary, black - adversarial direction. `a)` new samples are created by prioritizing areas where the model has the highest change in predictions, i.e. outside or close to the decision boundary. `b)` by forcing the network to be consistent with its predictions, the cluster gets naturally expanded, indirectly making input data closer to its center when compared to the edges of the cluster.

is important to note that combining conditional entropy and VAT often yields superior results compared to using either technique in isolation, especially in semi-supervised settings (Miyato et al., 2018; Oliver et al., 2018). However, entropy minimisation only ensures that the model is extremely confident about each data point, instead of placing decision boundaries in low-density areas (Oliver et al., 2018). As the number of labeled samples decreases, resulting in little information about true class distributions, the risk of approaching a solution where decision boundaries cut through the unlabeled data increases. This concern is further amplified by the fact that a potentially large, high-dimensional feature space is likely to be sparsely populated, which allows for more incorrectly placed boundaries that satisfy the aim of entropy minimisation algorithms. We tested our hypothesis by enabling conditional entropy during training and observed no improvement in the overall results.

4.1.3 Feature normalisation

We found that standardizing the input feature vectors individually yields better results, as opposed to commonly used batch normalization (Ioffe and Szegedy, 2015), or no normalization at all. At first glance, it might seem counter-intuitive, as features hold sensitive semantics about their corresponding samples, and any type of normalisation that does not take other statistics into consideration imposes the risk of harming the relationships between dimensions. However, this is done only as a pre-processing step, computing the standard score for vectors that are produced by the ResNet-101 model. Indeed, this is no different to widely popular image normalization (Shu et al., 2018; Laine and Aila, 2016), as G_r was not trained to output task-driven features. Therefore, we convert each input into zero-mean unit variance:

$$\hat{X}_i = \frac{\hat{X}_i - \mu(\hat{X}_i)}{\sigma(\hat{X}_i)}. \tag{4.11}$$

4.1.4 Iterative refinement training

Decision-boundary iterative refinement training with a teacher (Shu et al., 2018) is a secondary training phase that disables the source signal in order to further minimize the error in the target domain. That leaves conditional entropy together with VAT, and introduces a consistency cost, to ensure that decision boundaries of a student model stay close to those of the teacher (Tarvainen and Valpola, 2017). Shu et al. (2018) note that their approach suffers from high variance in results, and sometimes might lead to a degenerate solution, that is mainly caused by entropy minimisation, which we discussed in Section 4.1.2. Instead, we propose evaluating on an exponential moving average of the network weights with decay 0.999. This idea is borrowed from the semi-supervised and generative fields of computer vision (Sohn et al., 2020; Karras et al., 2019), and is proven to consistently provide better results than the original (student) model, due to smooth weight updates. Although we see merit in disabling the source signal and having a secondary training phase, that is rather a potential avenue for future work than focus of this Chapter.

Table 4.1: The summary of adaptation paths. For the amount of videos, the numbers indicate how many samples there are on each side of the path (e.g.: 1438 train videos for UCF_f).

Path	Classes #	Train videos #	Test videos #
UCF_s-Olympic	6	601-250	240-54
UCF_s-$HMDB_s$	5	482-350	189-150
UCF_f-$HMDB_f$	12	1438-840	571-360

4.2 Datasets

The datasets chosen for evaluation are UCF101 (Soomro et al., 2012), HMDB51 (Kuehne et al., 2011), and Olympic Sports Dataset (Niebles et al., 2010). All of them comprise of clips that were collected from public databases, such as YouTube, Google videos, and Prelinger archive. The videos show humans in motion, therefore, there is a significant overlap between categories. A detailed summary of adaptation paths formed by the datasets is presented in Table 4.1, while snapshot examples of the basketball/shoot ball class are shown in Figure 4.3. UCF_f-$HMDB_f$ (f for "full") is an extended version of UCF_s-$HMDB_s$ (s for "small"), which is designed to include extra samples and overlapping categories, making it a more challenging adaptation path than its predecessor (Chen et al., 2019a).

4.3 Experiments

The changes we made to the TA^3N implementation are as follows:

1. Switched stochastic gradient descent optimiser to Adam;
2. Disabled learning rate and weight decay;
3. Added VAT regularisation;
4. Added z-score normalisation to the general-purpose features, $\hat{X}$;
5. Performed evaluation on the exponential moving average model.

Figure 4.3: Snapshots of the three chosen datasets, from the overlapping category basketball/shoot ball.

To keep the comparison with the initial implementation fair, we left all the hyperparameters and evaluation protocols the same (Chen et al., 2019a). We also use the same pre-extracted features, available at TA^{3}N repository[1]. We note that results reported in the TA^{3}N paper differ from the ones available at the repository, therefore, we report the latter, as that is the code we worked with. For the other approaches, we drew the results from their original papers. Earlier in this Chapter, we hypothesised that using virtual adversarial training in feature space could bring advantages in the unsupervised video domain adaptation setting. To test this theory, we conducted experiments on six different adaptation scenarios:

- **UCF$_\mathbf{s}$ - Olympic**, denoted by $U_s \rightarrow O$ and $O \rightarrow U_s$;
- **UCF$_\mathbf{s}$ - HMDB$_\mathbf{s}$**, denoted by $U_s \rightarrow H_s$ and $H_s \rightarrow U_s$;
- **UCF$_\mathbf{f}$ - HMDB$_\mathbf{f}$**, denoted by $U_f \rightarrow H_f$ and $H_f \rightarrow U_f$;

[1] `https://github.com/cmhungsteve/TA3N`

Table 4.2: The video classification accuracy (%) comparison with other methods on publicly available unsupervised video domain adaptation benchmark datasets.

Method	$U_s \rightarrow O$	$O \rightarrow U_s$	$U_s \rightarrow H_s$	$H_s \rightarrow U_s$	$U_f \rightarrow H_f$	$H_f \rightarrow U_f$
Sultani and Saleemi (2014)	33.33	47.91	68.70	68.67	-	-
Xu et al. (2016)	87.00	75.00	82.00	82.00	-	-
GFK (Jamal et al., 2018a)	84.65	86.44	89.53	95.36	-	-
SA (Jamal et al., 2018a)	83.92	86.07	90.25	94.40	-	-
DAAA (Jamal et al., 2018a)	91.60	89.96	-	-	-	-
TA^{3}N (Chen et al., 2019a)	98.15	84.58	98.00	**98.94**	73.05	77.23
TA^{3}N + VAT (Section 4.1)	**100.0**	**90.00**	**98.67**	98.42	**79.73**	**84.07**

The results are available in Table 4.2. For 5 out of 6 paths, the suggested changes improved upon the state of the art at the time. For $H_s \rightarrow U_s$, they maintain competitive performance. The most notable enhancements can be observed on the UCF_f-$HMDB_f$ dataset, where we decrease the error from 26.95% and 22.77% to 20.27% and 15.93% on $UCF_f \rightarrow HMDB_f$ and $HMDB_f \rightarrow UCF_f$ paths, respectively. Our approach achieved 100% accuracy in the $UCF_s \rightarrow$Olympic setting, and is very close to perfect performance on both $UCF_s \rightarrow HMDB_s$ and $HMDB_s \rightarrow UCF_s$ too. The above suggests that many modern video domain adaptation datasets are not sophisticated enough to provide a challenge for modern algorithms. We address this issue and introduce new SVW-UCF dataset in the next Section.

While working on this Chapter, we considered doing an ablation study, which could have added more insight to the presented research. However, we decided not to conduct one for this thesis. Our choice was to keep our main focus as discussed in the earlier chapters. Nevertheless, the idea of an ablation study remains interesting, and it could be a direction for future research.

4.4 SVW-UCF Dataset for Unsupervised Video Domain Adaptation

One could argue that whereas image-based domain adaptation problems include various degrees of domain discrepancy, many video data paths do not present the same level of

complexity. In fact, a sophisticated neural network architecture can achieve good results even without any adaptation mechanism present in the method itself, as demonstrated later in Table 4.5. The above leads to believe that the lack of challenging problems is one of the main reasons as to why video-based DA has not received as much attention as its counterpart. In this Section, we introduce a large-scale video-based domain adaptation dataset called SVW-UCF (Safdarnejad et al., 2015; Soomro et al., 2012), which presents a larger domain gap and consists of 25 overlapping categories. We evaluate both TA^{3}N Chen et al. (2019a) and our method presented in Section 4.1 on SVW-UCF, and empirically prove that SVW→UCF and UCF→SVW paths are more challenging than those which were available to public at the time of writing[2].

4.4.1 Dataset description

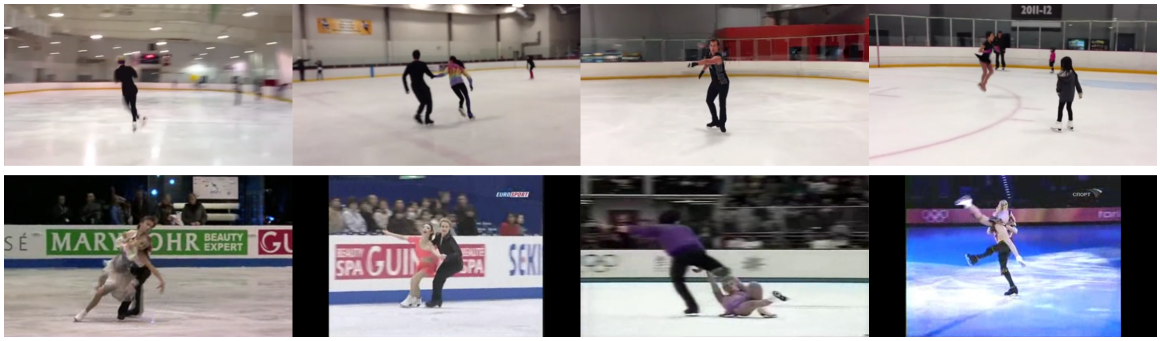
To build the dataset, we collected 25 distinct classes that are present in both SVW (Safdarnejad et al., 2015) and UCF101 (Soomro et al., 2012), listed in Table 4.3. For UCF101, we coupled some of the categories that are similar to each other into one, such as Basketball and Basketball Dunk to form Basketball, or Breaststroke and Front Crawl to form Swimming. In total, SVW-UCF consists of 5878 training and 2410 testing videos, shared between two domains (Table 4.4). For consistency, we followed the suggested SVW and UCF evaluation protocols[3,4], and chose train/test splits №1 for both.

The visual domain gap between SVW and UCF101 mostly comes in the differences between skill levels of sportsmen and camera operators in the videos. Whereas UCF101 mainly consists of clips that were filmed in their natural category environments that might provide additional cues to a classifier, e.g., basketball court or baseball pitch with teams and viewers, it is not the case with SVW. The latter dataset was captured solely with smartphones by amateurs, and contains more casual scenes, such as playing football in a backyard or

[2]We are aware of the Kinetics-Gameplay dataset (Chen et al., 2019a), however, it was released only in feature vector format, which significantly limits its usability in research.

[3]`http://cvlab.cse.msu.edu/project-svw.html`

[4]`https://www.crcv.ucf.edu/data/UCF101.php`



(a) Figure skating

(b) Hammer throw

(c) Punch

Figure 4.4: Snapshots of different categories from the SVW-UCF dataset. For each class, the samples from SVW and UCF101 are in the top and bottom row, respectively.

Table 4.3: Collected categories for SVW-UCF.

SVW-UCF	SVW	UCF101
Archery	Archery	Archery
Baseball	Baseball	Baseball Pitch
Basketball	Basketball	Basketball, Basketball Dunk
Biking	BMX	Biking
Boating	Rowing	Rowing, Rafting, Kayaking, Skijet
Bowling	Bowling	Bowling
Discus throw	Discusthrow	Throw Discus
Diving	Diving	Diving
Figure skating	Skating	Ice Dancing
Golf	Golf	Golf Swing
Gymnastics	Gymnastics	Floor Gymnastics
Hammer throw	Hammer throw	Hammer Throw
High jump	High jump	High Jump
Javelin throw	Javelin	Javelin Throw
Long jump	Long jump	Long Jump
Pole vault	Polevault	Pole Vault
Punching	Boxing	Boxing-Punching Bag, Boxing-Speed Bag, Punch
Shot put	Shotput	Shotput
Skiing	Skiing	Skiing
Soccer	Soccer	Soccer Juggling, Soccer Penalty
Swimming	Swimming	Breaststroke, Front Crawl
Tennis	Tennis	Tennis Swing
Volleyball	Volleyball	Volleyball Spiking
Weight lifting	Weight lifting	Bench Press, Clean and Jerk, Lunges
Wrestling	Wrestling	Sumo Wrestling

Table 4.4: Comparison of video domain adaptation datasets. We provide a pair of numbers to describe each side of the path (e.g.: 2466 training videos for SVW).

	UCF-Olympic	UCF-HMDB$_s$	UCF-HMDB$_f$	SVW-UCF
Average length (sec.)	5.4-10.6	5.8-3.3	7.2-3.3	15.5-6.6
Classes	6	5	12	25
Train samples	601-250	482-350	1438-840	2466-3412
Test samples	240-54	189-150	571-360	1057-1353

practicing field penalties on an empty pitch. The gap is further expanded by variations in camera angles and vibrations, lighting, and different orientations, which are not present in previously proposed datasets.

From the domain adaptation perspective, it is important to learn semantic information about actions with minimum bias to their visual appearance. For example, both bench press and lunges with dumbbells are considered to be weight lifting, although these exercises look very different from each other. Consequently, activities that represent same actions in a different manner introduce an additional yet positive challenge for this field of research. Hence, we grouped some of the classes that meet these criteria where it was possible, for example, wrestling and sumo wrestling, gymnastics and floor gymnastics. We also note that SVW-UCF has the largest amount of overlapping categories and unique samples, as well as greater average video length when comparing to the existing datasets. The details can be found in Table 4.4.

4.4.2 Comparison with other datasets

In this Subsection, we first present a set of potential reasons as to why existing video domain adaptation datasets do not present enough challenges for modern algorithms. Then, to test our hypotheses, we perform an evaluation of accuracy gaps between adaptation paths.

UCF-Olympic (Jamal et al., 2018b) and UCF-HMDB$_{small}$ (Sultani and Saleemi, 2014) were initially designed to have visually similar categories, while the main goal of domain adaptation is to learn semantics of an object or an action, regardless of its appearance (Ganin

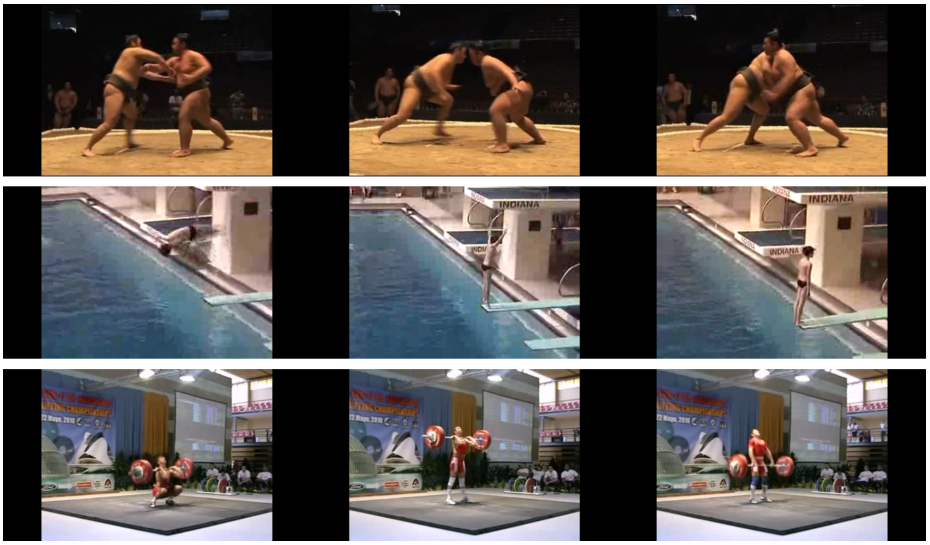


Figure 4.5: Snapshots of videos from the UCF101 dataset. Each frame belongs to a different sample, although it is clear that they are just small parts of a single, larger video.

and Lempitsky, 2015). Indeed, the absence of labels alone without the visual gap present turns this into a semi-supervised learning problem, where the target domain can simply be treated as a portion of unlabelled samples. To ensure that there is a sufficient domain discrepancy in SVW-UCF, we selected datasets that are different in both visual and semantic sense.

Neural network models are very sensitive to inputs, and require large amounts of training points to generalise and develop strong, reliable decision boundaries on the data manifold. All the previous datasets underdeliver in that regard, on average having less than 100 videos per class, per domain. This problem is further amplified by samples from UCF101, where many groups of clips are comprised of single large videos, and arguably cannot be considered as fully individual, distinct samples, as they share a lot of similar visual properties, as shown in Figure 4.5. We address this issue by providing 331.5 clips per class on average (versus second largest 267.4 for UCF-HMDB$_{full}$ (Chen et al., 2019a)), as well as greater video length to ensure a significant number of frames in SVW-UCF (Table 4.4). Clearly, there are methods that were specifically created to aid the requirements of neural networks in

Table 4.5: Accuracy (%) gap comparison between different video domain adaptation paths, using TA^{3}N model. U→O stands for UCF→Olympic, U→H$_s$ for UCF→HMDB$_{small}$, U→H$_f$ for UCF→HMDB$_{full}$, S→U for SVW→UCF, and vice versa.

	U→O	O→U	U→H$_s$	H→U$_s$	U→H$_f$	H→U$_f$	S→U	U→S
Source only	93.82	84.58	94.40	92.61	71.67	73.91	60.55	54.41
Target only	100.0	99.41	98.67	98.94	82.78	94.92	89.10	91.45
Accuracy gap	6.18	14.83	4.27	6.33	11.11	21.01	28.55	37.04

cases where the amount of training data is limited (Miyato et al., 2018; Yun et al., 2019), however, that is rather the focus of other research fields, such as unsupervised and semi-supervised learning. When it comes to domain adaptation, the purpose is to address the problem of the lack of data in the target domain, however, on condition that the source domain has sufficient task-related information (Ganin and Lempitsky, 2015).

To test whether aforementioned properties are important in domain adaptation datasets and deep learning algorithms, we trained the TA^{3}N model in one domain setting, i.e., in a standard, supervised manner, with adaptation mechanisms disabled. Obtained results (Table 4.5) suggest that the field of video DA needs to shift towards problems with greater visual and semantic discrepancies, as well as more classes and samples, since a strong CNN backbone model alone is already enough to achieve good performance on a large portion of available paths. In addition to that, we also trained and tested the two best performing video DA methods at the time: TA^{3}N (Chen et al., 2019a) and TA^{3}N+VAT (Section 4.1) on SVW-UCF, the results can be found in Table 4.6. Even with adaptation in place, the difference between the highest accuracy and 'Target only' is still significant (14.45% and 25.51% for SVW→UCF and UCF→SVW, respectively), which opens up opportunities for future research. When it comes to individual category performance (Figure 4.6), we found that the lowest scores were obtained in cases where the domain gap mainly consists of visual discrepancies. On UCF frames, classes such as Basketball, High jump, Hammer throw, Javelin throw, Punching, and Shot put mostly have professional pitch/stadium setups and crowd in the background, while many SVW videos of these sports do not share the same

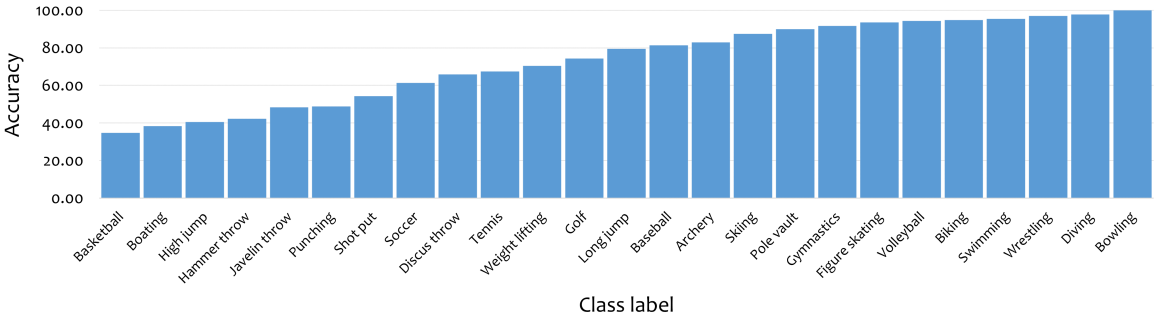


Figure 4.6: Individual category results of TA^{3}N+VAT (Section 4.1) on the UCF→SVW adaptation path, sorted by accuracy (%).

Table 4.6: Video domain adaptation algorithms evaluation on SVW-UCF.

Method	SVW→UCF	UCF→SVW
Source only	60.55%	54.41%
TA^{3}N (Chen et al., 2019a)	67.85%	59.98%
TA^{3}N+VAT (Section 4.1)	74.65%	65.94%
Target only	89.10%	91.45%

properties. On the other hand, activities that do not possess the same degree of freedom when it comes to the place where they can be performed in (e.g., skating always requires an ice rink), generally have higher accuracy: Skiing, Gymnastics, Figure skating, Swimming, Diving, and Bowling. This leads to a conclusion that video DA algorithms still heavily rely on visual cues, while the actions themselves are rather complementary. The above is further supported by a very strong performance on the Wrestling class - even though the domains have slightly different sports constructing the category (Table 4.3), the presence of the arena makes it easy for the classifier to assign the correct prediction. We also note that the addition of virtual adversarial training in feature space, which we described in Section 4.1, helps to achieve better performance when compared to just the TA^{3}N baseline on the adaptation paths that were introduced in this Section.

4.5 Discussion

In this Chapter, we proposed several approaches for unsupervised video domain adaptation: virtual adversarial training in feature space as well as normalization of input features, and testing on the exponential moving average weights. We empirically show that when combined together, these techniques significantly improve the performance of the recent state of the art model called TA^{3}N (Chen et al., 2019a). The proposed changes are independent of each other, easy to implement, applicable to other deep learning tasks, do not add much computational overhead, and can be combined with most existing neural network architectures. We stress that performing VAT on features is faster than performing VAT on individual frames, as it does not interfere with the feature extraction parts of the system. This property opens up additional opportunities for methods that work with pre-extracted features and/or have multiple networks, such as encoder/decoder models.

We also presented SVW-UCF - a large-scale dataset for video domain adaptation, which contains more categories and delivers a greater domain discrepancy, when compared to existing datasets at the time of writing. We evaluated the accuracy gap between all available adaptation paths and found that most of them do not present enough challenge for modern techniques, as good performance can be achieved by simply using a robust neural network model. Finally, we tested two state of the art video DA algorithms on SVW-UCF and looked at the results of the TA^{3}N+VAT method on UCF→SVW in detail. The latter suggests that the field of unsupervised video DA needs to shift towards problems with greater visual gaps, as that is the area where current methods struggle the most.

Virtual adversarial training is a method used for model regularisation, a practice that falls under the umbrella of data augmentation. While working on this Chapter and reading the relevant literature, we noticed the lack of research in the area of temporal augmentation. This gap, which stood out prominently, naturally prompted us to explore this uncharted territory. In the upcoming Chapter, we will introduce a set of innovative data augmentation

techniques specially tailored for video data. These techniques will be applied with the aim of enhancing the performance of video action recognition tasks. The exploration of temporal augmentation techniques not only contributes to the current understanding but also opens up a promising path for future research in this field. The insights gained in this Chapter serve as a foundational step towards further investigations.

5 Extending Temporal Data Augmentation for Video Action Recognition

Due to its ability of expanding and populating the training distribution through synthetically created samples, pixel space augmentation was successfully used as the main driver in a number of semi-supervised (Zhang et al., 2017; Yun et al., 2019; Miyato et al., 2018; Sohn et al., 2020), self-supervised (Chen et al., 2020; He et al., 2020; Misra and Maaten, 2020), and domain adaptation (French et al., 2017; Shu et al., 2018; Mao et al., 2019) studies. The use of feature space augmentation was also explored for both static and sequential imagery (Chu et al., 2020; DeVries and Taylor, 2017a; Liu et al., 2018a; Gorpincenko et al., 2021b), yielding improvements in models' accuracy. Data augmentation for videos, however, still remains an under-explored research area, as most works have been treating inputs as stacks of static images rather than temporally linked series of data.

Kim et al. (2020b) have shown that the time domain consideration while designing augmentations can be superior to its spatial-only variants for video recognition. In this Chapter, we expand on the work done by Kim et al. (2020b). We extend their proposed techniques even further to fully utilise the time domain and achieve a deeper level of temporal perturbations, which results in more accurate and robust classifiers. The contributions of this Chapter can be summarised as follows:

1. We expand the list of available augmentations in RandAugment-T (discussed in Section 2.6.1) by adding three new augmentations that are video-specific and are done within a single sample;

2. We increase the amount of magnitude checkpoints for all augmentation techniques to allow for non-linear temporal perturbations;

3. We propose to linearly change the bounding pox positions for cut-and-paste algorithms, such as CutOut, CutMix, and CutMixUp, and their extensions, as well as the mixing ratio in MixUp and CutMixUp extensions (discussed in Section 2.6.1);

4. The recognition results of the aforementioned techniques on the UCF101 (Soomro et al., 2012) and the HMDB51 (Kuehne et al., 2011) datasets either maintain competitive or exceed performance achieved by the previous work.

5.1 Method

In this Section, we first introduce the three video-specific augmentations. Then, we propose a framework that introduces controlled magnitude perturbations for simple transformations. Finally, we change the nature of a number of delete, cut-and-paste and blend algorithms from static to dynamic.

5.1.1 Single video augmentation

RandAugment (Cubuk et al., 2020) is an automated data augmentation framework that randomly selects a number of transformations for a given image. From a list of K operations, RandAugment takes N augmentations with the magnitude of M. Each transformation has a probability of $\frac{1}{K}$ to be chosen. A total of $K = 14$ operations are presented: Identity, Rotate, Posterise, Equalise, Sharpness, Translate-X, Translate-Y, Colour, AutoContrast, Solarise, Contrast, Brightness, Shear-X, and Shear-Y.

RandAugment-T (Kim et al., 2020b) introduces M_1 and M_n, two magnitude points that are placed at the start and the end of each video. This allows for smooth augmentation transitions across the frames and brings the temporal component to the equation, where possible. The work also extends the list of available transformations by including Colour-Invert, albeit it having static magnitude. All the operations mentioned above are taken

Figure 5.1: Proposed single video augmentations. Unaugmented video, VideoReverse, FrameFadeIn, and VideoCutMix in the 1st, 2nd, 3rd, and 4th row, respectively.

directly from image augmentation, and are applied to a single video. Operations such as Identity, AutoContrast, Equalise, and ColourInvert do not have varying M, and hence are applied evenly across the sample.

Although previous work sticks to the aforementioned list of transformations (Cubuk et al., 2018, 2020; Ho et al., 2019; Lim et al., 2019), the purpose of this paper is to propose temporal augmentations, rather than suggest a new augmentation policy. Therefore, we expand the list of available operations by introducing VideoReverse, FrameFadeIn, and VideoCutMix - transformations that are designed specifically for video samples, which are shown in Figure 5.1. VideoReverse turns the video backwards, creating a rewind effect, yet maintaining the semantics and integrity of the sample. FrameFadeIn is inspired by FadeMixUp (Kim et al., 2020b), with the main difference being the use of a single sample and a simpler mixing ratio calculation:

$$\tilde{x_t} = (1 - \lambda_t)x_t + \lambda_t x_{n-t}, \tag{5.1}$$

where $\tilde{x}$, x, n, and λ indicate the mixed data, original data, total number of frames, and mixing ratio, respectively. Unlike FadeMixUp, we do not sample start and end points for λ interpolation. Instead, we gradually increase it from 0 to 0.5 until the middle of the video,

then decrease it back to 0:

$$\lambda_t = \begin{cases} \frac{t}{n}, & \text{if } t \leq \frac{n}{2}, \\ \frac{n-t}{n}, & \text{otherwise.} \end{cases} \tag{5.2}$$

This maintains a healthy trade-off between spatial and temporal perturbations - when the distance between frames is large, the mixing ratio is small, and vice versa. Although it is possible to use sampled magnitudes instead, it significantly increases the risk of breaking temporal consistency. VideoCutMix is a temporal extension of CutMix (Yun et al., 2019; Kim et al., 2020b) that can be applied to a single video:

$$\tilde{x_t} = M \odot x_t + (1 - M) \odot \hat{x_t}, \tag{5.3}$$

where M, $\hat{x}$, and $\odot$ denote the binary region mask indicating where to drop out or fill in from two separate frames, video with randomly shuffled frames, and element-wise multiplication, respectively. Although cut-and-pasting happens within the same sample, the nondeterministic nature introduces a certain risk of altering data to the point where semantics may be significantly damaged or lost. To keep it to minimum, we set the region ratio to 0.2 of the original frame size and keep the position of the bounding box static. As with all single sample augmentations, labels remain unchanged in VideoReverse, FrameFadeIn, and VideoCutMix.

5.1.2 MagAugment

RandAugment-T (Kim et al., 2020b) implements augmentation transitions across frames by putting two magnitude checkpoints, M_1 and M_n, at the start and the end of samples, and calculating the other M_t via linear interpolation. The introduced change in magnitude leads to better video action recognition performances, when compared to its static variant (Kim et al., 2020b). Our hypothesis is that having more magnitude checkpoints placed along the sample results in greater generalisation performance, as they are more likely to mimic perturbations observed in real-life conditions. Phenomena such as flashes, sudden camera

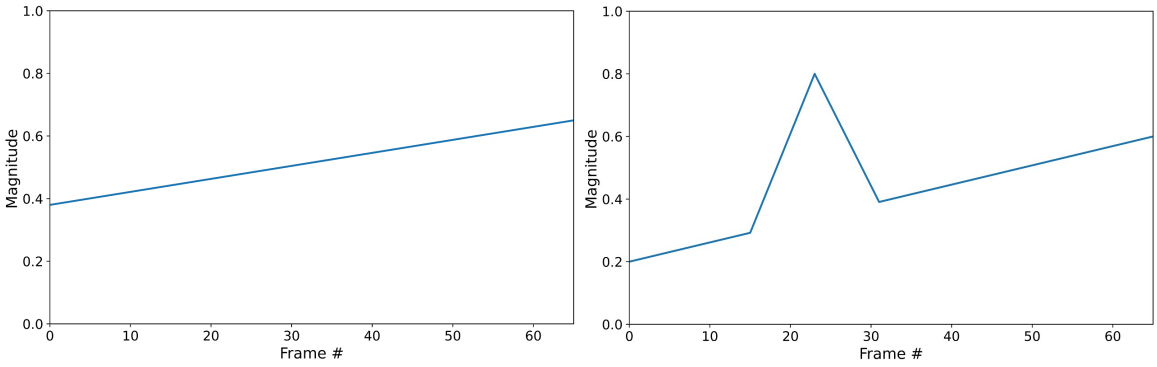


Figure 5.2: Visual comparison of magnitude manipulations proposed in RandAugment-T (Kim et al., 2020b) (left) and MagAugment (right). For MagAugment, β was set to 8.

shaking and/or movement, loss of focus, and exposure adjustments tend to happen in much shorter time periods than the length of the entire video. In this Subsection, we propose MagAugment (Figure 5.2) - a framework designed to increase the magnitude diversity even further, without interrupting the temporal consistency.

We start with the linear signal connecting the two ends of the magnitude array. To introduce short and sporadic magnitude swings, we sample a point from the uniform distribution, $M_p \sim U(M_{min}, M_{max})$, where the parameters represent the minimum and maximum magnitude values for a given transformation. The duration of the perturbations in frames is set to $j \sim U(1, \beta)$, where β is the MagAugment parameter. Finally, the location of the point is drawn from $p \sim U(1 + j, n - j)$, where n is the total amount of frames. The process can be repeated to model several fluctuations. To incorporate the magnitude swings into the original signal, we linearly interpolate from M_{p-j} to M_p, then back to M_{p+j}. As a result, the overall augmentation direction is maintained, while allowing for occasional, more aggressive changes in pixel space that do not necessarily follow the general trend.

5.1.3 Temporal deleting, cut-and-pasting, and blending

The temporal adaptations of CutOut (DeVries and Taylor, 2017b) and CutMix (Yun et al., 2019) apply a bounding box, B, to every frame of a given sample, without changing its

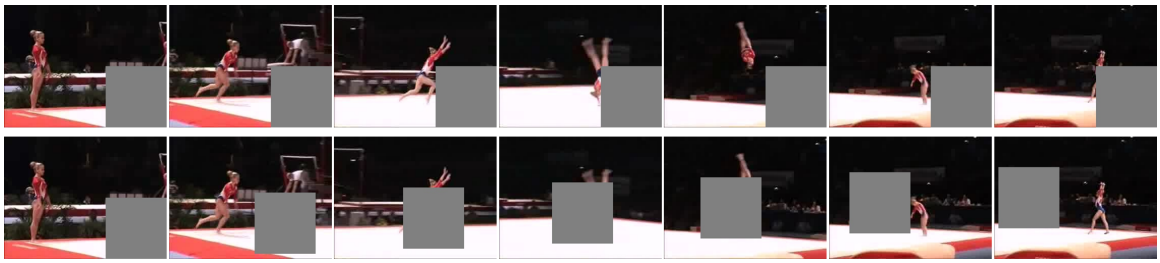
position, r. In CutMix, the frame sequences are also aligned with the video used for mixing. The concept of static location is practiced in the algorithms' extensions as well - CubeCutOut, CubeCutMix, CutMixUp, and CubeCutMixUp (Kim et al., 2020b). The temporal version of MixUp (Zhang et al., 2017) has a fixed mixing ratio, λ, and remains so in its extensions too - CutMixUp, FrameCutMixUp, CubeCutMixUp (Yoo et al., 2020; Kim et al., 2020b). Such an idea removes the stochastic behaviour that would be introduced if the aforementioned augmentations were applied to frames separately, without acknowledging them as a part of data series. However, the regularisation techniques themselves can be temporally varied too. By taking a deterministic approach, we are able to enhance the level of spatiotemporal augmentations and involve more bounding box positions and mixing ratios within a batch.

In this Subsection, we propose dynamic r and λ, by linearly changing them across the time dimension. The concept is similar to RandAugment-T, only this time we generate r_1/r_n or λ_1/λ_n instead of magnitude points for the start and the end of a training sample. Therefore, for delete and cut-and-paste algorithms r becomes:

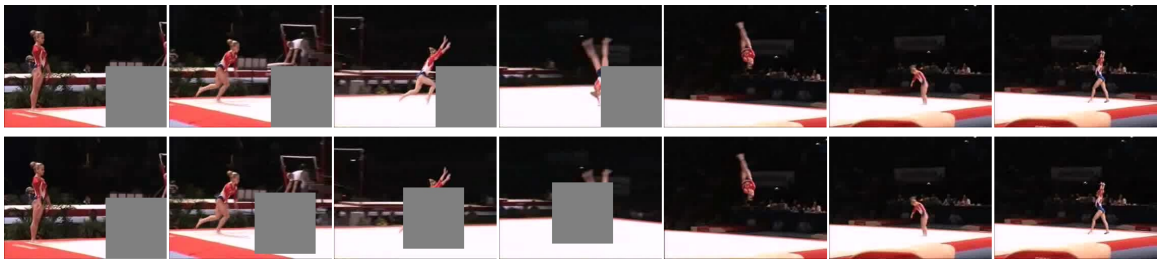
$$\begin{gathered} r_{1_x} \sim U(r_w, W - r_w), r_{n_x} \sim U(r_w, W - r_w), r_w = W\sqrt{1 - I}, \\ r_{1_y} \sim U(r_h, H - r_h), r_{n_y} \sim U(r_h, H - r_h), r_h = H\sqrt{1 - I}, \\ I \sim \text{Beta}(\alpha, \alpha), \end{gathered} \tag{5.4}$$

where W, H, U, and α are the frame width, frame height, uniform distribution, and beta distribution parameter, respectively. The dynamic r ensures that the bounding box changes its location smoothly as the video goes on. Please note that unlike the previous implementations (DeVries and Taylor, 2017b; Yun et al., 2019; Yoo et al., 2020; Kim et al., 2020b), we ensure that the bounding box is fully within the frame at all times, therefore guaranteeing label consistency across the time dimension and omitting the label recalculation step. r_w and r_h are calculated once per video. When start and end points are found, the rest is computed via linear interpolation between the two. Although the authors of FadeMixUp

(Kim et al., 2020b) introduced dynamic λ, CutMixUp, FrameCutMixUp, CubeCutMixUp still used the static one. To make the three algorithms temporally varied when it comes to blending, we substitute MixUp with FadeMixUp. All of the above results in seven new regularisation approaches: FloatCutOut, FloatCubeCutOut, FloatCutMix, FloatCubeCutMix, FloatCutMixUp, FloatCubeCutMixUp, and FloatFrameCutMixUp, which are shown in Figure 5.3.



(a) Top: CutOut (DeVries and Taylor, 2017b; Kim et al., 2020b), Bottom: FloatCutOut



(b) Top: CubeCutOut (Kim et al., 2020b), Bottom: FloatCubeCutOut

(c) Top: CutMix (Yun et al., 2019; Kim et al., 2020b), Bottom: FloatCutMix

Figure 5.3: Visual comparison of temporal deleting, cut-and-pasting, and blending algorithms with static and dynamic r and λ.

(d) Top: CubeCutMix (Kim et al., 2020b), Bottom: FloatCubeCutMix

(e) Top: CutMixUp (Kim et al., 2020b), Bottom: FloatCutMixUp

(f) Top: CubeCutMixUp (Kim et al., 2020b), Bottom: FloatCubeCutMixUp

(g) Top: FrameCutMixUp (Kim et al., 2020b), Bottom: FloatFrameCutMixUp

Figure 5.3: Visual comparison of temporal deleting, cut-and-pasting, and blending algorithms with static and dynamic r and λ (continued).

5.2 Experiments

We train and test the approaches mentioned in this paper on the UCF-101 (Soomro et al., 2012) and HMDB51 (Kuehne et al., 2011) datasets to assess their effectiveness. The UCF101 dataset contains 13 320 videos split into 101 categories, whereas HMDB-51 consists of 6 766 videos split into 51 categories. To keep the comparison with the previous work fair, we use the same training and testing splits, network architecture (Feichtenhofer et al., 2019), optimiser (Kingma and Ba, 2014), training setup and hyperparameters, and additional techniques, such as learning rate warm-up (Goyal et al., 2017a) and cosine learning rate scheduling (Loshchilov and Hutter, 2016). Please note that for methods proposed by Kim et al. (Kim et al., 2020b), we report results achieved by running the published code[1] ourselves. In all tables, **bold text** indicates the highest accuracy. For the UCF101, the displayed numbers represent the results on the 1$^{\text{st}}$ VIPriors action recognition challenge split. For the HMDB51, we report the average results obtained from 3 different splits (Kuehne et al., 2011).

5.2.1 Single video augmentation

In this Subsection, we evaluate VideoReverse, FrameFadeIn, VideoCutMix, and MagAugment, the results can be found in Table 5.1. RandAugment indicates static magnitude, applied evenly to all the frames of a given video. For RandAugment-T+, $M1$ and M_n are set to $M - \delta$ and $M + \delta$, respectively, where $\delta = U(0, 0.5 * M)$, and M comes from the values used by RandAugment. RandAugment-T++ stands for the extended version desribed in Section 5.1.1, and includes VideoReverse, FrameFadeIn and VideoCutMix. For MagAugment that we introduced in Section 5.1.2, a grid search of $\beta \in [2, 4, 8, 16]$ and the amount of magnitude checkpoints, $P \in [1, 2, 3, 4]$, was used to obtain the highest accuracy, with $\beta = 8$ and $P = 2$ demonstrating the best performance. We apply MagAugment to all transformations present in RandAugment-T++, apart from the ones that cannot facilitate

[1] `https://github.com/taeoh-kim/temporal_data_augmentation`

Table 5.1: Video action recognition accuracy (%) results on the UCF101 and HMDB51 datasets for single video augmentation techniques. To save space, we abbreviate RandAugment as RA.

Method	UCF Top1	UCF Top5	HMDB Top1	HMDB Top5
Baseline	54.93	77.43	39.12±0.41	69.89±0.40
RA (Cubuk et al., 2020)	69.82	88.57	49.24	79.94
RA-T+ (Kim et al., 2020b)	70.47	89.94	49.48±0.58	80.17±0.42
RA-T++ (Section 5.1.1)	70.74	90.04	49.60±0.44	80.21±0.36
MagAugment (Section 5.1.2)	**72.18**	**93.78**	**50.70±0.62**	**81.12±0.59**

varying magnitude - Identity, Reverse, AutoContrast, Equalise, ColourInvert, FrameFadeIn, and VideoCutMix. To obtain the baseline results, we use the same SlowFast model (Feichtenhofer et al., 2019) with all augmentations disabled. To save space, we abbreviate RandAugment as RA.

The results confirm that including more single video augmentations provides a benefit with no additional computational overhead, thanks to the nature of RandAugment. By including three new transformations, we were able to increase the gap attained by RandAugment-T from 0.65% and 0.24% to 0.92% and 0.36% in UCF101 Top1 and HMDB51 Top1 settings, respectively, when compared to spatial-only RandAugment. This shows that designing video-specific augmentations can be more valuable than applying single image operations to video samples, albeit with varying magnitude. By enabling MagAugment, we increase the gap further to 2.36% and 1.46%, in UCF101 Top1 and HMDB51 Top1 settings, respectively. This suggests that simulating real-life perturbations by implementing aggressive yet controlled magnitude swings can provide a much greater benefit than a careful transformation selection.

We also include an ablation study for RandAugment-T++. To see whether each of the three proposed augmentations is useful in the video action recognition setting, we disable them one by one, leaving the remaining two untouched. The results can be found in Table 5.2. It is clear that without VideoReverse or FrameFadeIn, the RandAugment-T++ framework falls short of its best results. The exclusion of VideoCutMix does provide a small boost

Table 5.2: Ablation study accuracy (%) results on the UCF101 and HMDB51 datasets for single video augmentation techniques proposed in Section 5.1.1. Please note that to save space, we abbreviate VideoReverse, FrameFadeIn and VideoCutMix as VR, FFI, and VCM, respectively.

Method	UCF Top1	UCF Top5	HMDB Top1	HMDB Top5
Baseline	54.93	77.43	39.12±0.41	69.89±0.40
RA-T+ (Kim et al., 2020b)	70.47	89.94	49.48±0.58	80.17±0.42
RA-T++ without VR	70.52	89.94	49.50±0.42	80.10±0.37
RA-T++ without FFI	70.58	89.98	49.58±0.45	80.06±0.36
RA-T++ without VCM	**70.76**	**90.12**	49.54±0.47	80.17±0.39
RA-T++ (Section 5.1.1)	70.74	90.04	**49.60±0.44**	**80.21±0.36**

in accuracy on the UCF101 dataset, however, that comes at the cost of performance on the HMDB51. We hypothesise that the stochastic nature of VideoCutMix brings in some instabilities when it comes to the training process, and might not be beneficial for all datasets. Perhaps, a clearer trend could be observed given more training time, however, that is rather a potential avenue for future work rather than focus of this Chapter.

5.2.2 Temporal deleting, cut-and-pasting, and blending

We present the results of Cutout (DeVries and Taylor, 2017b), CutMix (Yun et al., 2019), and CutMixUp (Yoo et al., 2020), and their temporal extensions (Kim et al., 2020b), which can be found in Table 5.3. We omit references in this table to save space. We prefix our methods with F to save space and indicate floating bounding box positions and mixing ratios. The highest accuracies in each respective group are indicated with **bold** font. Single video augmentation is turned off in this experiment. Although the CutOut variants struggle to beat the baseline and the CutMix spin-offs demonstrate a rather small boost in accuracy, it is clear that having dynamic ratio and lambda helps the model to consistently achieve better performance. The floating extensions of CutOut show an average gain of 2.97% and 1.92% in UCF101 Top1 and HMDB51 Top1 settings, respectively, when compared to their static variants. When it comes to CutMix, the observed improvements in the same settings are 1.49% and 2.65%. For CutMixUp, the gains are 2.19% and 2.99%. FloatFrameCut-

Table 5.3: Video action recognition accuracy (%) results on the UCF101 and HMDB51 datasets for temporal deleting, cut-and-pasting, and blending techniques. References are omitted to save space. Our methods that we presented in Section 5.1.3 are prefixed with *F*. Each group of algorithms is separated with a double line.

Method	UCF Top1	UCF Top5	HMDB Top1	HMDB Top5
Baseline	54.93	77.43	39.12±0.41	69.89±0.40
CutOut	51.16	74.25	36.93±0.48	68.07±0.46
CubeCutOut	51.82	76.73	37.50±0.39	68.53±0.34
*F*CutOut	54.24	76.23	39.02±0.48	69.89±0.38
*F*CubeCutOut	**54.68**	**77.19**	**39.25±0.37**	**70.00±0.38**
CutMix	53.03	76.78	34.69±0.50	65.67±0.43
CubeCutMix	54.91	77.34	36.75±0.46	67.23±0.41
*F*CutMix	55.25	77.27	37.32±0.51	68.24±0.45
*F*CubeCutMix	**55.66**	**78.00**	**39.42±0.42**	**69.98±0.41**
CutMixUp	60.08	82.14	43.13±0.40	74.19±0.36
CubeCutMixUp	60.16	82.14	43.15±0.40	74.24±0.34
FrameCutMixUp	61.02	82.97	42.88±0.46	74.08±0.35
*F*CutMixUp	62.41	84.59	45.06±0.43	75.78±0.35
*F*CubeCutMixUp	62.38	84.70	45.12±0.45	75.85±0.36
*F*FrameCutMixUp	**63.04**	**85.64**	**45.98±0.41**	**76.90±0.33**

MixUp scores the highest accuracy, improving over the baseline by 8.11% and 6.86% in UCF101 Top1 and HMDB51 Top1 settings, respectively. FloatCubeCutOut, FloatCubeCutMix, and FloatFrameCutMixUp perform the best in their respective groups, suggesting that retaining some of the frames of a video unaffected might yield additional benefits (Figure 5.3b, Figure 5.3d, and Figure 5.3g, respectively). The authors would also like to note that the floating extensions of all algorithms perform better than their static variants, when we perform side-by-side comparison, i.e., CutMixUp and *F*CutMixUp, CubeCutMixUp and *F*CubeCutMixUp, and so on.

5.3 Discussion

In this Chapter, we introduced several novel temporal data augmentation methods. We showed that developing video-specific transformations and including more aggressive magnitude transitions is beneficial for networks that aim to solve video action recognition.

We extended temporal versions of CutOut, CutMix, and CutMixUp further by changing their nature from static to dynamic. We compared our proposed techniques with their predecessor variants and observed an overall improvement in performance for all methods in Top1 and Top5 settings on the UCF101 and HMDB51 datasets. Potential future work avenues include combining single video augmentations with delete, cut-and-paste, and blend techniques to expand the total amount of possible augmentation combinations, covering more baseline models to analyse applicability and versatility of the proposed methods, testing the framework on larger datasets, such as Kinetics (Carreira and Zisserman, 2017) and Something-Something-v2 (Goyal et al., 2017b), and exploring the effectiveness of augmentations targeted specifically at background and/or foreground in video action recognition.

6 Conclusions and Future Work

In this thesis, we have made four main contributions. This Chapter concludes the discoveries, connects the aforementioned research topics, and discusses potential avenues for future work.

6.1 Contributions

Initially, the main goal of the thesis was to find ways to improve the classification performance and robustness of the JellyMonitor system. For this reason, we first addressed poor generalisation caused by data imbalance that was overlooked in the previous work (French et al., 2018). This work was brought to fruition in Chapter 3 (Gorpincenko et al., 2021a). Unfortunately, as the project was ceased for reasons independent of UEA, we had to slightly alter the direction of research. We still wanted to keep exploring different approaches of artificially expanding the training set for video-related tasks, which we did by incorporating virtual adversarial training in feature space in the unsupervised video domain adaptation setting in Chapter 4 (Gorpincenko et al., 2021b). While working on that topic, we noticed that many publicly available datasets do not present enough challenge for the modern video UDA algorithms. Therefore, we introduced two new adaptation paths formed by UCF101 and HMDB51, which contained more classes and offered greater domain discrepancy when compared to the existing adaptation paths at the time, that we describe in Chapter 4 (Gorpincenko and Mackiewicz, 2021). In Chapter 5, we expand on the earlier work (Kim et al., 2020b) by proposing novel ways to augment video samples in the video action recognition setting (Gorpincenko and Mackiewicz, 2022).

6.1.1 Improving automated sonar video analysis to notify about jellyfish blooms

In Chapter 3, we contributed an application of the generative adversarial networks to a real-world problem, which was rather novel back then, as GANs only started becoming popular. We also introduced an event classifier network that takes confidence scores of the frame classifier model as the input - something that we have not come across in the literature. The addition of simple yet effective techniques, such as weighted loss and confidence threshold, helped us to get closer to the desired accuracy on the jellyfish class, as well as significantly reduce the amount of false positives.

Future work

In hindsight, we could have experimented with video classification frameworks, however, back then our knowledge on the topic was limited, and out-of-the-box frameworks required a large amount of processing time and memory, which rendered them unfeasible for Jelly-Monitor usage. Nowadays, however, it would be possible to operate modern architectures even within the constraints imposed by the embedded platform. We hypothesise that the largest potential improvement can be achieved by updating the tracking algorithm such that it is more sensitive to the object being mostly faded at the end of sequences - the problem that we discussed at the end of Section 3.2.5 and showed in Figures 3.5 and 3.10. Of course, the classification and generative fields have advanced significantly since then, and replacing the frame classifier and data generation network backbones would most likely result in better performance.

6.1.2 Virtual adversarial training in feature space for unsupervised video domain adaptation

In Chapter 4, we contributed a novel use of virtual adversarial training in feature space, which proved to be effective in the unsupervised domain adaptation setting. We also pro-

vided our findings and discussion on entropy minimisation and iterative refinement training, and came to a conclusion that very similar results can be achieved with simpler substitutes, or no substitutes at all. Finally, we introduced a large video domain adaptation dataset that can be used in unsupervised and semi-supervised settings.

Future work

The idea of applying VAT in feature space is rather versatile - it can be used with any type of input data. Although we strongly believe that enforcing the cluster assumption in feature space can be useful in most semi-supervised or unsupervised tasks, we lack empirical data to support the claim. Therefore, one potential avenue for future work on this topic could be extending the experiments to other fields, such as single image semi-supervised learning or unsupervised domain adaptation.

6.1.3 Extending temporal data augmentation for video action recognition

In Chapter 5, we contributed three new single-sample video augmentations that we included in the existing framework RandAgument-T (Kim et al., 2020b). Then, we proposed MagAugment - a framework for creating controlled magnitude perturbations to simulate real-life phenomena, such as flashes, sudden camera shaking, loss of focus, and exposure adjustments. Finally, we changed the formulation of existing delete, cut-and-paste, and blend algorithms for videos to enhance the level of spatiotemporal augmentations, which resulted in seven novel regularisations. All of the proposed techniques yielded a boost in accuracy in the video action recognition setting, when compared to their predecessors.

Future work

Although our contributions resulted in better network performance, it is unclear whether the proposed techniques can be useful if applied to more sophisticated video classification networks, especially the family which utilises optical flow. It is possible that the introduced

perturbations can in fact harm the models that heavily rely on temporal structure and consistency across the frames. Therefore, we would like to test our work on a broad range of baseline models, as well as larger datasets, such as Kinetics (Carreira and Zisserman, 2017) and Something-Something-v2 (Goyal et al., 2017b). Moreover, we are yet to combine the regularisations described in Section 5.1.3 together, which opens up an interesting line of research on how to choose the best policy for such augmentations.

6.2 Conclusion

To conclude, this thesis has encompassed a diverse range of contributions aimed at improving automated sonar video analysis, enhancing video domain adaptation techniques, and extending temporal data augmentation for video action recognition. While the specific focus of each chapter varied, the common thread binding them together was the pursuit of more robust and effective solutions for video analysis tasks. Through addressing issues of lack of data, data imbalance, and domain shifts, as well as improving upon existing techniques and proposing innovative augmentations, we have made significant strides in our understanding of these areas. Each Chapter presented its own challenges and had its own unique solutions, leading to better data efficient deep learning algorithms for video processing systems. Looking ahead, the future work outlined in each section opens up promising avenues for further exploration and refinement. The collective impact of these contributions, when considered together, underscores our commitment to advancing the field of computer vision and the potential for these approaches to make valuable contributions to real-world applications and future research endeavors.

7 Bibliography

Able, K. W., Grothues, T. M., Rackovan, J. L., and Buderman, F. E. (2014). Application of mobile dual-frequency identification sonar (didson) to fish in estuarine habitats. *Northeastern Naturalist*, 21(2):192–209.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bachman, P., Alsharif, O., and Precup, D. (2014). Learning with pseudo-ensembles. *Advances in neural information processing systems*, 27.

Basodi, S., Ji, C., Zhang, H., and Pan, Y. (2020). Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3):196–207.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. J. V. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359.

Becker, A., Whitfield, A. K., Cowley, P. D., Järnegren, J., and Næsje, T. F. (2013). Potential effects of artificial light associated with anthropogenic infrastructure on the

abundance and foraging behaviour of estuary-associated fishes. *Journal of Applied Ecology*, 50(1):43–50.

Benaim, S., Ephrat, A., Lang, O., Mosseri, I., Freeman, W. T., Rubinstein, M., Irani, M., and Dekel, T. (2020). Speednet: Learning the speediness in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9922–9931.

Bennett, M., Becker, A., Gaston, T., and Taylor, M. (2020). Connectivity of large-bodied fish with a recovering estuarine tidal marsh, revealed using an imaging sonar. *Estuaries and Coasts*, 44:1579—-1587.

Bishop, C. M. (1995). Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116.

Boero, F., Brotz, L., Gibbons, M. J., Piraino, S., and Zampardi, S. (2016). 3.10 impacts and effects of ocean warming on jellyfish. *Explaining ocean warming: Causes, scale, effects and consequences*, pages 213–237.

Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992.*, pages 144–152.

Bothmann, L., Windmann, M., and Kauermann, G. (2016). Realtime classification of fish in underwater sonar videos. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 65(4):565–584.

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. (2017). Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731.

Braga, L., Giraldo, A., and Godinho, A. (2022). Evaluation of three methods for manually counting fish in dam turbines using didson. *Hydrobiologia*, 849:309—-321.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096.

Brown and Lowe (2003). Recognising panoramas. In *Proceedings Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1218–1225.

Burges, C. J. and Schölkopf, B. (1996). Improving the accuracy and speed of support vector machines. *Advances in neural information processing systems*, 9.

Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Chen, M.-H., Kira, Z., AlRegib, G., Yoo, J., Chen, R., and Zheng, J. (2019a). Temporal attentive alignment for large-scale video domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6321–6330.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2172–2180.

Chen, X., Wang, X., Changpinyo, S., Piergiovanni, A., Padlewski, P., Salz, D., Goodman, S., Grycner, A., Mustafa, B., Beyer, L., et al. (2022). Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*.

Chen, Y., Schmid, C., and Sminchisescu, C. (2019b). Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7063–7072.

Chu, P., Bian, X., Liu, S., and Ling, H. (2020). Feature space augmentation for long-tailed data. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 694–710, Cham. Springer International Publishing.

Cireşan, D., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). High-performance neural networks for visual object classification. *Computing Research Repository - CORR*.

Cireşan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, 13(1):21–27.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. (2021). Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pages 886–893.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

DeVries, T. and Taylor, G. W. (2017a). Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*.

DeVries, T. and Taylor, G. W. (2017b). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Ding, H., Jiang, X., Shuai, B., Liu, A. Q., and Wang, G. (2019). Semantic correlation promoted shape-variant context for segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8885–8894.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766.

Drucker, H., Schapire, R., and Simard, P. (1992). Improving performance in neural networks using a boosting algorithm. *Advances in neural information processing systems*, 5.

Edgington, D. R., Cline, D. E., Davis, D., Kerkez, I., and Mariette, J. (2006). Detecting, tracking and classifying animals in underwater video. In *OCEANS 2006*, pages 1–5.

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211.

Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941.

Fernando, B., Habrard, A., Sebban, M., and Tuytelaars, T. (2013). Unsupervised visual domain adaptation using subspace alignment. In *2013 IEEE International Conference on Computer Vision*, pages 2960–2967.

Frazao, X. and Alexandre, L. A. (2014). Dropall: Generalization of two convolutional neural network regularization methods. In *International Conference Image Analysis and Recognition*, pages 282–289. Springer.

French, G., Mackiewicz, M., and Fisher, M. (2017). Self-ensembling for visual domain adaptation. *arXiv preprint arXiv:1706.05208*.

French, G., Mackiewicz, M., Fisher, M., Challiss, M., Knight, P., Robinson, B., and Bloomfield, A. (2018). Jellymonitor: automated detection of jellyfish in sonar images using neural networks. In *2018 14th IEEE International Conference on Signal Processing (ICSP)*, pages 406–412.

French, G., Oliver, A., and Salimans, T. (2020). Milking cowmask for semi-supervised image classification. *arXiv preprint arXiv:2003.12022*.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3):121–136.

G Masilamani, J., S Jesudoss, K., Kanavillil, N., Satpathy, K., V K Nair, K., and Azariah, J. (2000). Jellyfish ingress: A threat to the smooth operation of coastal power plants. *Current science*, 79:567–569.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. (2018). Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Gong, B., Grauman, K., and Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Con-*

ference on Machine Learning, volume 28 of *Proceedings of Machine Learning Research*, pages 222–230, Atlanta, Georgia, USA. PMLR.

Gong, B., Sha, F., and Grauman, K. (2012a). Overcoming dataset bias: An unsupervised domain adaptation approach. In *NIPS Workshop on Large Scale Visual Recognition and Retrieval*, volume 3. Citeseer.

Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012b). Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gopalan, R., Li, R., and Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. In *2011 International Conference on Computer Vision*, pages 999–1006.

Gorpincenko, A., French, G., Knight, P., Challiss, M., and Mackiewicz, M. (2021a). Improving automated sonar video analysis to notify about jellyfish blooms. *IEEE Sensors Journal*, 21(4):4981–4988.

Gorpincenko, A., French, G., and Mackiewicz, M. (2021b). Virtual adversarial training in feature space to improve unsupervised video domain adaptation. *Electronic Imaging*, 2021(10):258–1–258–6.

Gorpincenko, A. and Mackiewicz, M. (2021). Svw-ucf dataset for video domain adaptation. In *Proceedings of the International Conference on Image Processing and Vision Engineering - IMPROVE*, pages 107–111. INSTICC, SciTePress.

Gorpincenko, A. and Mackiewicz, M. (2022). Extending temporal data augmentation for video recognition. In *Proceedings of the 37th International Conference on Image and Vision Computing New Zealand - IVCNZ*, pages 1–1. Springer.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017a). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al. (2017b). The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850.

Graham, W. M. (2001). Numerical increases and distributional shifts of chrysaora quinquecirrha (desor) and aurelia aurita (linné) (cnidaria: Scyphozoa) in the northern gulf of mexico. *Hydrobiologia*, 451(1):97–111.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Grote, A. B., Bailey, M. M., Zydlewski, J. D., and Hightower, J. E. (2014). Multibeam sonar (didson) assessment of american shad (alosa sapidissima) approaching a hydroelectric dam. *Canadian Journal of Fisheries and Aquatic Sciences*, 71(4):545–558.

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 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5769–5779.

Hayes, J., Melis, L., Danezis, G., and Cristofaro, E. (2017). Logan: Evaluating information leakage of generative models using generative adversarial networks. *arXiv preprint arXiv:1705.07663*.

Hayes, J. W., Hay, J., Maxwell, I., and Quarterman, A. (2015). Estimating trout abundance with cataraft-mounted dual-frequency identification sonar: a comparison with drift diving. *North American Journal of Fisheries Management*, 35(3):528–536.

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., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.

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.

Helminen, J. and Linnansaari, T. (2021). Object and behavior differentiation for improved automated counts of migrating river fish using imaging sonar data. *Fisheries Research*, 237:105883.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6629–6640.

Ho, D., Liang, E., Chen, X., Stoica, I., and Abbeel, P. (2019). Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr.

Hong, W., Wang, Z., Yang, M., and Yuan, J. (2018). Conditional generative adversarial network for structured domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510.

Hughes, J. B., Bentz, B., and Hightower, J. E. (2018). A non-invasive approach to enumerating white sturgeon (acipenser transmontanus richardson, 1863) using side-scan sonar. *Journal of Applied Ichthyology*, 34(2):398–404.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470.

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. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1(4):364–378.

Jamal, A., Namboodiri, V. P., Deodhare, D., and Venkatesh, K. (2018a). Deep domain adaptation in action space. In *BMVC*, volume 2, page 5.

Jamal, A., Namboodiri, V. P., Deodhare, D., and Venkatesh, K. (2018b). Deep domain adaptation in action space. In *BMVC*.

Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231.

Jing, D., Han, J., Wang, G., Wang, X., Wu, J., and Chen, G. (2016). Dense multiple-target tracking based on dual frequency identification sonar (didson) image. In *OCEANS 2016 - Shanghai*, pages 1–5.

Jing, D., Han, J., Wang, J., Wang, X., and Xu, Z. (2018). Three-dimensional distribution of fish using an imaging sonar. *Journal of Fisheries of China*, 42(6):996–1005.

Jones, R. E., Griffin, R. A., and Unsworth, R. K. (2021). Adaptive resolution imaging sonar (aris) as a tool for marine fish identification. *Fisheries Research*, 243:106092.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020a). Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. (2021). Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020b). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119.

Keefer, M. L., Caudill, C. C., Johnson, E. L., Clabough, T. S., Boggs, C. T., Johnson, P. N., and Nagy, W. T. (2017). Inter-Observer Bias in Fish Classification and Enumeration Using Dual-frequency Identification Sonar (DIDSON): A Pacific Lamprey Case Study. *Northwest Science*, 91(1):41–53.

Kim, D., Kim, H., Jung, S., Koo, J., , and Myung, H. (2015a). A vision-based detection algorithm for moving jellyfish in underwater environment. In *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 144–145.

Kim, D., Shin, J.-U., Kim, H., Kim, H., Lee, D., Lee, S.-M., and Myung, H. (2016). Development and experimental testing of an autonomous jellyfish detection and removal robot system. *International Journal of Control, Automation and Systems*, 14(1):312–322.

Kim, H., Kim, D., Shin, J., and Myung, H. (2015b). Development of a uav-type jellyfish monitoring system using deep learning. In *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 495–497.

Kim, H., Koo, J., Kim, D., Jung, S., Shin, J., Lee, S., and Myung, H. (2016). Image-based monitoring of jellyfish using deep learning architecture. *IEEE Sensors Journal*, 16(8):2215–2216.

Kim, J., Cha, S., Wee, D., Bae, S., and Kim, J. (2020a). Regularization on spatio-temporally smoothed feature for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12103–12112.

Kim, J.-Y. and Ha, J.-E. (2021). Spatio-temporal data augmentation for visual surveillance. *IEEE Access*, PP:1–1.

Kim, T., Lee, H., Cho, M., Lee, H. S., Cho, D. H., and Lee, S. (2020b). Learning temporally invariant and localizable features via data augmentation for video recognition. In *European Conference on Computer Vision*, pages 386–403. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). Hmdb: A large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563.

Lagarde, R., Peyre, J., Amilhat, E., Bourrin, F., Prellwitz, F., Simon, G., and Faliex, E. (2021). Movements of non-migrant european eels in an urbanised channel linking a mediterranean lagoon to the sea. *Water*, 13(6).

Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

Langkau, M. C., Balk, H., Schmidt, M. B., and Borcherding, J. (2012). Can acoustic shadows identify fish species? a novel application of imaging sonar data. *Fisheries Management and Ecology*, 19(4):313–322.

LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Lee, S., Park, B., and Kim, A. (2019). Deep learning based object detection via style-transferred underwater sonar images ∗∗this work is supported through a grant from msip (no 2015r1c1a2a01052138), iitp grant funded by msit (no.2017-0-00067), and a grant from endowment project of kriso (pes9390). authors are grateful to sonartech for sharing sample videos for the research. *IFAC-PapersOnLine*, 52(21):152–155. 12th

IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019.

Lemley, J., Bazrafkan, S., and Corcoran, P. M. (2017). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869.

Liebelt, J. and Schmid, C. (2010). Multi-view object class detection with a 3d geometric model. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 1688–1695.

Lim, S., Kim, I., Kim, T., Kim, C., and Kim, S. (2019). Fast autoaugment. *Advances in Neural Information Processing Systems*, 32.

Link, J. (2006). Widespread and persistent increase of ctenophora in the continental shelf ecosystem off ne usa. *Marine Ecology-progress Series - MAR ECOL-PROGR SER*, 320:153–159.

Liu, B., Wang, X., Dixit, M., Kwitt, R., and Vasconcelos, N. (2018a). Feature space transfer for data augmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, L., Lu, H., Cao, Z., and Xiao, Y. (2018b). Counting fish in sonar images. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3189–3193.

Llewellyn, L. E., Bainbridge, S., Page, G., O'Callaghan, M. D., and Kingsford, M. J. (2016). Stingercam: A tool for ecologists and stakeholders to detect the presence of venomous tropical jellyfish. *Limnology and Oceanography: Methods*, 14(10):649–657.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Long, M., Wang, J., Ding, G., Sun, J., and Yu, P. S. (2013). Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207.

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Magowan, K., Reitsma, J., and Murphy, D. (2012). Use of dual-frequency identification sonar to monitor adult river herring in a small coastal stream. *Marine and Coastal Fisheries*, 4(1):651–659.

Malej, A., Kogovšek, T., Ramšak, A., Catenacci, L., et al. (2012). Blooms and population dynamics of moon jellyfish in the northern adriatic. *CBM-Cahiers de Biologie Marine*, 53(3):337.

Mao, F., Wu, X., Xue, H., and Zhang, R. (2018). Hierarchical video frame sequence representation with deep convolutional graph network. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Mao, X., Ma, Y., Yang, Z., Chen, Y., and Li, Q. (2019). Virtual mixup training for unsupervised domain adaptation. *arXiv preprint arXiv:1905.04215*.

Martin-Abadal, M., Ruiz-Frau, A., Hinz, H., and Cid, Y. (2020). Jellytoring: Real-time jellyfish monitoring based on deep learning object detection. *Sensors*, 20(6):1708.

Matsuura, F. and Fujisawa, N. (2007). Detection and removal of jellyfish using underwater image analysis. *Journal of Visualization - J VIS*, 10:259–260.

Mescheder, L. M., Geiger, A., and Nowozin, S. (2018). Which training methods for gans do actually converge? In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 3478–3487.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks. *CoRR*, abs/1611.02163.

Mills, C. E. (2001). Jellyfish blooms: are populations increasing globally in response to changing ocean conditions? *Hydrobiologia*, 451(1):55–68.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Misra, I. and Maaten, L. v. d. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Niebles, J. C., Chen, C.-W., and Fei-Fei, L. (2010). Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision*, pages 392–405. Springer.

Oh, S., Marks, R., and El-Sharkawi, M. (1991). Query based learning in a multilayered perceptron in the presence of data jitter. In *Proceedings of the first international forum on applications of neural networks to power systems*, pages 72–75. IEEE.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. (2018). Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31.

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

Park, C.-D., Lee, K., Kim, S.-H., and Fujimori, Y. (2012). Performance of a conical jellyfish exclusion device installed in a trawl net. *Fisheries Science*, 78:23–32.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR.

Pauly, D., Graham, W., Libralato, S., Morissette, L., and Deng Palomares, M. L. (2009). Jellyfish in ecosystems, online databases, and ecosystem models. *Hydrobiologia*, 616(1):67–85.

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.

Piergiovanni, A., Fan, C., and Ryoo, M. (2017). Learning latent subevents in activity videos using temporal attention filters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29.

Purcell, J., Uye, S.-i., and Lo, W. (2007). Anthropogenic cause of jellyfish blooms and their direct consequences for humans: A review. *Marine Ecology Progress Series*, 350:153–174.

Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*.

Ranjan, A. and Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170.

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.

Reed, R., Oh, S., Marks, R., et al. (1992). Regularization using jittered training data. In *International Joint Conference on Neural Networks*, volume 3, pages 147–152.

Rife, J. and Rock, S. M. (2003). Segmentation methods for visual tracking of deep-ocean jellyfish using a conventional camera. *IEEE Journal of Oceanic Engineering*, 28(4):595–608.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Russo, P., Carlucci, F. M., Tommasi, T., and Caputo, B. (2018). From source to target and back: Symmetric bi-directional adaptive gan. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8099–8108.

Safdarnejad, S. M., Liu, X., Udpa, L., Andrus, B., Wood, J., and Craven, D. (2015). Sports videos in the wild (svw): A video dataset for sports analysis. In *Proc. International Conference on Automatic Face and Gesture Recognition*, Ljubljana, Slovenia.

Saito, K., Ushiku, Y., and Harada, T. (2017). Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997. PMLR.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, page 901.

Sankaranarayanan, S., Balaji, Y., Castillo, C. D., and Chellappa, R. (2018a). Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8503–8512.

Sankaranarayanan, S., Balaji, Y., Jain, A., Lim, S. N., and Chellappa, R. (2018b). Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3752–3761.

Shen, J., Qu, Y., Zhang, W., and Yu, Y. (2018). Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Shen, W., Zhu, Z., Zhang, J., Cao, Z., and Peng, Z. (2020). Fish target recognition and counting based on dual-frequency identification sonar. *Fishery Modernization*, 47(6):81–87.

Shiganova, T. A. (1998). Invasion of the black sea by the ctenophore mnemiopsis leidyi and recent changes in pelagic community structure. *Fisheries Oceanography*, 7(3-4):305–310.

Shu, R., Bui, H. H., Narui, H., and Ermon, S. (2018). A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*.

Sietsma, J. and Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–79.

Simard, P., Steinkraus, D., and Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 958–963.

Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 568–576, Cambridge, MA, USA. MIT Press.

Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: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. *Advances in neural information processing systems*, 33:596–608.

Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild.

Spampinato, C., Chen-Burger, Y.-H., Nadarajan, G., and Fisher, R. (2008). Detecting, tracking and counting fish in low quality unconstrained underwater videos. In *Proc. 3rd Int. Conf. on Computer Vision Theory and Applications (VISAPP)*, volume 2, pages 514–519.

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.

Stark, M., Goesele, M., and Schiele, B. (2010). Back to the future: Learning shape models from 3d CAD data. In *British Machine Vision Conference, BMVC 2010, Aberystwyth, UK, August 31 - September 3, 2010. Proceedings*, pages 1–11.

Sultani, W. and Saleemi, I. (2014). Human action recognition across datasets by foreground-weighted histogram decomposition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 764–771.

Sultani, W. and Saleemi, I. (2014). Human action recognition across datasets by foreground-weighted histogram decomposition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 764–771.

Summers, C. and Dinneen, M. J. (2019). Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270. IEEE.

Sun, B. and Saenko, K. (2014). From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *British Machine Vision Conference, BMVC 2014, Nottingham, UK, September 1-5, 2014*.

Sun, B. and Saenko, K. (2015). Subspace distribution alignment for unsupervised domain adaptation. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 24.1–24.10. BMVA Press.

Sun, L., Jia, K., Yeung, D., and Shi, B. E. (2015). Human action recognition using factorized spatio-temporal convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4597–4605, Los Alamitos, CA, USA. IEEE Computer Society.

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., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016a). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer*

Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 2818–2826.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016b). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tarling, P., Cantor, M., Clapés, A., and Escalera, S. (2022). Deep learning with self-supervision and uncertainty regularization to count fish in underwater images. *PLOS ONE*, 17(5):1–26.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30.

Terayama, K., Shin, K., Mizuno, K., and Tsuda, K. (2019). Integration of sonar and optical camera images using deep neural network for fish monitoring. *Aquacultural Engineering*, 86:102000.

Tian, Y., Yan, Y., Zhai, G., Guo, G., and Gao, Z. (2022). Ean: event adaptive network for enhanced action recognition. *International Journal of Computer Vision*, 130(10):2453–2471.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

Tulyakov, S., Liu, M., Yang, X., and Kautz, J. (2018). Mocogan: Decomposing motion and content for video generation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1526–1535.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

van Hal, R., Griffioen, A., and van Keeken, O. (2017). Changes in fish communities on a small spatial scale, an effect of increased habitat complexity by an offshore wind farm. *Marine Environmental Research*, 126:26–36.

Varol, G., Laptev, I., and Schmid, C. (2017). Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1510–1517.

Vasile, G., d'Urso, G., Goarant, A., de Oliveira, E., and Lungu, E. (2016). Potential of active ultrasound monitoring systems for jellyfish detection. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–4.

Vázquez, D., López, A. M., and Ponsa, D. (2012). Unsupervised domain adaptation of virtual and real worlds for pedestrian detection. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*, pages 3492–3495.

Vinyals, O., Jia, Y., Deng, L., and Darrell, T. (2012). Learning with recursive perceptual representations. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I.

Walther, D., Edgington, D. R., and Koch, C. (2004). Detection and tracking of objects in underwater video. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA. PMLR.

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, L., Tong, Z., Ji, B., and Wu, G. (2021). Tdn: Temporal difference networks for efficient action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1895–1904.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. (2016). Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer.

Wei, Y., Duan, Y., and An, D. (2022). Monitoring fish using imaging sonar: Capacity, challenges and future perspective. *Fish and Fisheries*, 23(6):1347–1370.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. (2022). Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR.

Wu, Z., Wang, Z., Wang, Z., and Jin, H. (2018). Towards privacy-preserving visual recognition via adversarial training: A pilot study. In *Proceedings of the European conference on computer vision (ECCV)*, pages 606–624.

Xiang, S. and Li, H. (2017). On the effects of batch and weight normalization in generative adversarial networks. *arXiv preprint arXiv:1704.03971*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Xu, M., Pérez-Rúa, J.-M., Escorcia, V., Martinez, B., Zhu, X., Zhang, L., Ghanem, B., and Xiang, T. (2021). Boundary-sensitive pre-training for temporal localization in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7220–7230.

Xu, T., Zhu, F., Wong, E. K., and Fang, Y. (2016). Dual many-to-one-encoder-based transfer learning for cross-dataset human action recognition. *Image and Vision Computing*, 55:127–137. Handcrafted vs. Learned Representations for Human Action Recognition.

Yoo, J., Ahn, N., and Sohn, K.-A. (2020). Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*.

Yuan, L., Chen, D., Chen, Y.-L., Codella, N., Dai, X., Gao, J., Hu, H., Huang, X., Li, B., Li, C., et al. (2021). Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*.

Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., and Choe, J. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zhang, C., Zou, Y., Chen, G., and Gan, L. (2020). Pan: Towards fast action recognition via learning persistence of appearance. *arXiv preprint arXiv:2008.03462*.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2019). Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008.

Zhou, H., Llewellyn, L., Wei, L., Creighton, D., and Nahavandi, S. (2015). Marine object detection using background modelling and blob analysis. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 430–435.

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*.

Zhu, Q., Yeh, M.-C., Cheng, K.-T., and Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1491–1498.

Zou, Y., Yu, Z., Kumar, B., and Wang, J. (2018). Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305.