JellyMonitor: automated detection of jellyfish in sonar images using neural networks

Geoff French*, Michal Mackiewicz*, Mark Fisher*, Mike Challiss†, Peter Knight†, Brian Robinson† and Angus Bloomfield‡
*School of Computing Sciences, University of East Anglia, Norwich, UK
Email: g.french@uea.ac.uk, m.mackiewicz@uea.ac.uk, m.fisher@uea.ac.uk
†Cefas Technology Ltd., Cefas, Lowestoft, UK
‡EDF Energy R&D UK

***Abstract*—JellyMonitor is an self-contained automated system that detects jellyfish blooms and reports their presence. It uses an embedded platform to analyse sonar imagery captured by a sonar imaging device. The software utilises a combination of classic computer vision techniques and deep neural networks to detect and classify objects captured by the sonar imaging device. We report on the development of this system and present results obtained from deploying a prototype.**

Keywords: jellyfish, sonar, deep neural networks, image classification, object detection

I. Introduction

Coastal nuclear power stations use seawater for cooling purposes and incur significant cost when jellyfish blooms clog the seawater inlets. With sufficient advance warning, these costs can be significantly reduced. JellyMonitor is designed to be a self-contained system that uses an embedded platform to analyse images captured by a sonar imaging device in order to detect jellyfish blooms and report their presence, providing advance warning of their approach.

The software component of JellyMonitor is designed to analyse sonar imagery in real-time while operating within the computational constraints of an embedded platform. It uses classic computer vision techniques to detect and track potential objects of interest after which a deep neural network classifier is used to identify them. This process is complicated by the large quantities of noise present in sonar imagery. We report on the development of the software component of JellyMonitor system and present results acquired from the deployment of a prototype.

This paper is organised as follows: in section II we cover the hardware component of JellyMonitor; in section III we describe our data set; in section IV we discuss the image processing software in JellyMonitor; in section V we present our results and finally we present our conclusions in section VI.

II. Hardware

The end goal of the JellyMonitor project is to develop a system that can be deployed offshore and can detect and report jellyfish blooms via satellite or other telemetry uplink. It is intended to consist of an underwater rig tethered to a floating buoy. The sonar device, battery and processing hardware are housed in a module attached to the underwater rig while the buoy houses the transmitter. A prototype deployment consisting of an underwater rig and the sonar capture device can be seen in Figure 1. The system must be capable of operating autonomously for two to three months – the length of a typical jellyfish season – once deployed. This necessitates strict power usage constraints as the battery must be able to sustain the system for the deployment period.

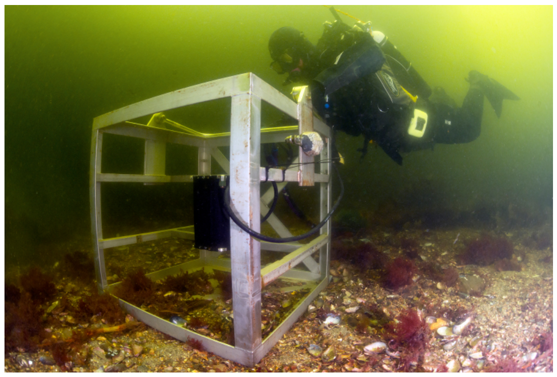


Fig. 1. The underwater rig with capture device, deployed in Oban, Scotland. Image courtesy of Dr. Martin Sayer at the National Facility for Scientific Diving at the Scottish Association for Marine Science (SAMS).

The use of 3MHz sonar to detect and image moon jellyfish was reported by Han *et al.* [1] in 2009. These results were confirmed by experiments with the Sound Metrics ARIS Explorer 3000 sonar imaging device [2].

Many computer vision algorithms – particularly those that utilise deep learning – are computationally intensive. This necessitates the use of a high performance embedded computing platform that is able to process the incoming sonar footage in real-time while utilising less than 20W of power. The nVidia Jetson TX2 [3] is currently (as of writing) the best choice given these constraints. Its small physical footprint combined with its ability to execute deep neural network based image classifiers quickly were essential for our system.

III. Data

The data used to develop and test our approach was captured in 3 locations over the course of the project; within an enclosed tank, within Lowestoft harbour and on a coastal site in Oban, Scotland.

The tank based footage was captured at the beginning of the project by Cefas and Cefas Technology Ltd. staff. The controlled environment and the presence of captive jellyfish allowed the capture device settings to be optimised for acquiring clear sonar images of moon jellyfish. Furthermore, it generated large quantities of footage in which jellyfish were known to be present and could usually be easily seen.

Later footage was captured by placing the sonar device underwater within Lowestoft Harbour and releasing jellyfish within its field of view. Towards the end of the project, footage more representative of real life conditions was captured by placing the capture device within an underwater rig on the seabed off the coast of Oban, Scotland. This resulted in footage of jellyfish in the wild and would be more representative of real life conditions.

A breakdown of the footage obtained is shown in Table I.

Capture location	Period	Time (days, H:MM:SS)	# of frames
Tank	2015	8:39:07	186,164
Lowestoft Harbour	2015	3:22:53	180,277
Seabed survey	2015	4:33:20	134,778
Oban, Scotland	2016	11:55:55	598,926
Oban, Scotland	2017	27d, 7:14:14	14,119,741

TABLE I
SONAR FOOTAGE OVERVIEW

IV. AUTOMATED SONAR IMAGE ANALYSIS

A. Sonar imagery

The ARIS Explorer 3000 captures sonar images in a polar co-ordinate system. Each 'pixel' has a `bin, beam` co-ordinate with its value representing the strength of the signal reflected back to the device. The `beam` identifies the angle while the `bin` identifies the range / distance. The ARIS Explorer 3000 has an angular field of view of 29.5° split into 128 beams. It has a maximum range of 15m. This is however adjustable; the sonar footage captured over the course of the JellyMonitor project had a maximum range of 10m or less (jellyfish were visible at a range of up to 5m). The number of bins depends on the distance chosen by the user. The sonar images take the form of 8-bit per pixel greyscale images that represent accoustic intensity in the range of 0 to 80dB.

B. Challenges

1) Images in polar vs cartesian co-ordinate frames: Processing sonar images without first converting them to a cartesian co-ordinate frame is challenging as the appearance of objects varies – mainly in terms of aspect ratio – depending on their distance from the sonar capture device. While the apparent size of an object in the distance axis will remain constant, it will vary in the angular axis as it will occupy a larger angular segment the closer it is to the capture device. This problem can be addressed by converting the image to a cartesian co-ordinate frame.

Convering a polar-space sonar image to a cartesian co-ordinate frame and processing the complete image can incur a high computational load due to the size – in pixels – of the resulting cartesian space image. An example image can be seen in the left panel of Figure 5. The area of the image is proportional to the square of the maximum range of the polar image; the range is projected along the vertical axis, while the width of the of the radial arc traced out is proportional to the maximum range. At a resolution of 5mm per pixel, a sonar image with a distance range of 0.7m to 10m would require a 800×1864 pixel image in a cartesian co-ordinate frame. Processing a complete image of this size using a neural network is essentially infeasible on all but the most powerful of desktop GPUs due to the large resolution. Furthermore, the polar to cartesian image conversion itself can be expensive at such resolutions due to the per-pixel bilinear filtering operations involved in this process.

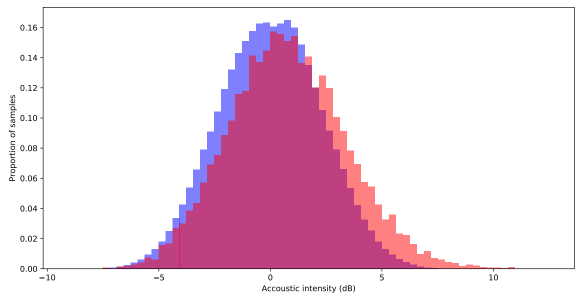


Fig. 2. Signal vs noise: histogram of accoustic intensity values found in a 0.05m radius region surrounding a faint object (red) vs empty background (blue).

2) Noise: There is a great deal of noise inherent in sonar imagery. Object detection, tracking and identification in such noisy conditions is one of the major hurdles that must be overcome.

In order to quantify the noise present in our dataset, we found a faint object that we intended for our system to be able to detect and track in footage captured in Oban. We extracted pixel values from a 0.05m radius circular region of footage over 33 frames centred on the faint object, resulting in 11,814 pixel values, which were converted to accoustic intensity in dB. We contrast this with an approximately rectangular region of 0.88m x 0.6m over 130 frames of empty footage, consisting of 2,688,000 accoustic intensity values. Histograms of these are shown in Figure 2. While the pixel values generated by the capture device measure accoustic intensity in decibels we adopt an image processing variant of signal to noise ratio (SNR), given the domain in which we are operating. Our SNR is shown in Table II. Please note that background subtraction (see section IV-C1) is applied prior to computing the values shown.

C. Approach

In order to work within the computational power available, we developed a multi-stage algorithm that first attempts to quickly locate potential objects of interest and track their motion across the field of view of the sonar device. These

TABLE II

SIGNAL VS NOISE; μ_{bg} AND σ_{sig} ARE THE MEAN AND STANDARD DEVIATION OF THE ACCOUSTIC INTENSITY OF BACKGROUND PIXELS, μ_{sig} AND σ_{sig} ARE THE MEAN AND STANDARD DEVIATION OF THE ACCOUSTIC INTENSITY OF SIGNAL PIXELS AND SNR IS THE SIGNAL TO NOISE RATIO. FILTERED PIXELS ARE OBTAINED BY APPLYING GAUSSIAN FILTERING TO THE DEVICE SPACE IMAGE WITH $\sigma = 6$.

	μ_{bg}	σ_{bg}	μ_{sig}	σ_{sig}	SNR ($\frac{\mu_{sig}}{\sigma_{bg}}$)
Raw pixels	0.003	2.286	0.789	2.623	0.343
Filtered	0.003	0.152	0.46	0.2	3.014

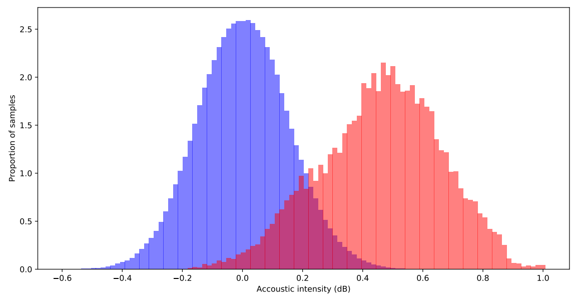


Fig. 3. Signal vs noise histogram of filtered accoustic intensity.

objects are later passed to a neural network classifier to identify them.

1) Background removal: Fixed structures are visible in some of the sonar footage, particularly that from the tank and harbour footage (see section III). Given that later components of the system detect bright regions for the purpose of locating objects of interest, removing fixed background structures simplifies their task. We maintain a per-pixel running mean of the previous 50 frames of footage and subtract this from the current frame. This is performed on the polar-space image.

2) Blob detection: The first step of detecting objects of interest is blob detection. We detect blobs in polar-space images in order to avoid the need to convert the images to cartesian-space first. Our blob detection algorithm is very simple; we first perform Gaussian filtering with $\sigma = 6$ pixels in polar-space to suppress the noise inherent in sonar images (see section IV-B2). After filtering we locate local maxima whose value exceed a threshold value of 0.58dB. The locations of the detections are converted to cartesian space as a scale of 1 unit to 1 metre.

Filtering the polar-space images increases the SNR from 0.343 to 3.014, as seen in Table II. The effect is illustrated in Figure 3.

The values for the filter radius and the threshold were determined by manual experimentation in order to ensure that faint objects can be detected – paying attention to the faint object mentioned in section IV-B2 – while keeping false detections to a minimum. We experimented with non-uniform filter shapes and filters whose radius varied with the distance from the sonar device in order to account for the varying horizontal scale factor across the polar space image. We found that the best performamnce was achieved when the filter averages over a consistent number of samples.

The detections are filtered in order to remove blobs that are closer than a minimum separation distance (0.05m); for any pair of detections that are too close to one another, the detection with the lowest brightness in the underlying image is discarded.

Note that the filtered image is only used for blob detection as the filtering smooths away the detail and visual cues required for identifying the object later in the pipeline.

3) Object tracking and patch extraction: Subsequent to detection, objects are tracked through multiple frames of footage. Given that the tracker receives a list of detection co-ordinates for the current frame from the blob detector, it must attempt to match each detection with one of the objects currently being tracked.

We use a Kalman filter [4] to predict the location in the current frame of all objects currently being tracked, given their motion history in previous frames. The predicted locations are matched with detections using the Hungarian algorithm [5]. Any matches where the distance between the predicted location and the detection is greater than a tracking error threshold of 0.13m are discarded.

When a tracked object has been visible for less than 4 frames, we do not consider there to be sufficient data to initialise Kalman filter. When there are less than 4 successful detections available we use the location of the last successful detection as the predicted position in the current frame.

The challenging conditions present in sonar footage result in objects frequently failing to be detected. To overcome this, we allow tracked objects to be absent for runs of up to 10 frames or absent in up to 80% of frames over which they are tracked, whichever is less. Objects absent for longer are considered to have disappeared from view and tracking is ceased. When tracking ceases, the motion path considered to represent a tracked object is passed to the next stage of the tracking process.

We discard objects that were tracked for less than 7 frames – incuding frames in which they are absent – as early experience showed that objects tracked for short periods of time most frequently resulted from false positive detections. We then apply Kalman smoothing to the sequence of observed positions – with gaps for frames where the object was absent – to generate a smooth path for the tracked object.

The blob detection threshold value of 0.58dB mentioned in section IV-C2 had to be chosen carefully, as too many spurious detections will result in legitmate detections from the previous frame being matched to spurious detections in the current frame, causing the object tracker to fail.

A sequence of 48x48 pixel (24cm) image patches centred on the object's location (drawn from the smoothed path) in each frame are extracted for classification in the final step.

4) Object classification: The object patches from the previous step are passed to a deep neural network (DNN) image classifier. A class probability vector is generated for each frame in which the object is visible. The predicted class

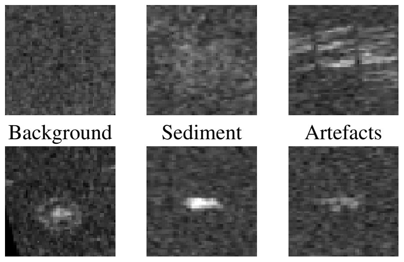


Fig. 4. Object class example images

probabilites are averaged across frames to produce a final classification result.

Our classifier is trained to distinguish among the variety of objects and phenomena that may trigger the blob detection and object tracking system, besides the jellyfish that are the focus of the system. We describe them below and illustrate them in Figure 4.

Background: noise within the imagery can overcome the threshold and cause a false detection. These objects are assigned to the background class.

Sediment: diffuse clouds of sediment arising from water turbulence

Artefacts: Strong reflections from the ocean surface can cause ringing artefacts. They have a distinctive pattern.

Jellyfish: Moon jellyfish have a distinctive central core surrounded by a ring. Their pulsating motion is also a strong cue.

Fish: Fish have a fairly distinctive shape. Their swimming motion is also a strong cue.

Seaweed: Drifts with ocean current much like jellyfish, although their shape is less uniform. This class is the most easily confused with jellyfish.

D. Training data

Deep neural networks (DNNs) have achieved state of the art results in image classification in recent years. It is for this reason that we chose to use them for the purpose of classifying objects of interest that are detected in the sonar footage. This accuracy however comes at the cost of large quantities of labelled training data.

Our DNN classifier identifies the contents of image patches extracted from the sonar data. Its training set consists of image patches containing various kinds of object extracted from the sonar footage.

Manually locating and annotating objects – with a particular focus on jellyfish – on a frame-by-frame basis was the approach used to develop our training set early in the project. This approach was satisfactory for footage acquired in a tank due to the constrained motion of the jellyfish and their relative abundance. This did not hold for ocean footage that by nature would be most representative of intended use case scenarios, when jellyfish presentations can be infrequent and rare. As a consequence they can easily be missed by human annotators. Once located, annotating their position in each frame in which they are present is a very laborious and time consuming task. Manually analysing the volume of footage acquired over the course of the project (see Table I) would not be achievable in a reasonable amount of time.

After developing the blob detection and tracking technique described in sections IV-C2 and IV-C3 we were able to use them grow our training set far more rapidly by automatically detecting and tracking objects of interest, thereby alleviating manual labellers of the task of finding objects and annotating their frame-by-frame position. By assigning a ground truth classification to an automatically tracked object (this is done using a labelling tool described in section IV-D1) they add multiple training images – one per frame – along with a corresponding ground truth class to the training set. This approach has the added benefit that the training set will consist of patches that will be more similar to those seen at interence time in the field.

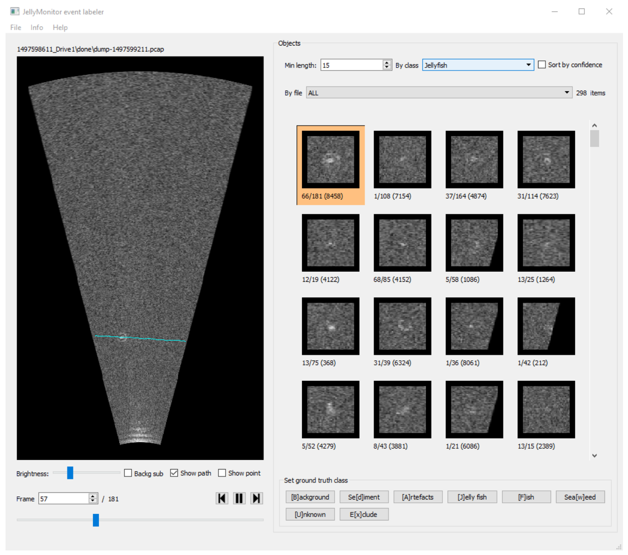
1) Labelling tool: A labelling tool (see Figure 5) was developed that allows a user to browse the objects found by the automatic detection and tracking system and assign ground truth classifications to them.

The 16 panes in the right side of the user interface (UI) show videos of detected objects. Each frame of video is a patch extracted from the corresponding frame of the sonar footage centred on the objects' position. Selecting a pane causes a full view of the original footage in which the object was detected to be displayed in the large pane on the left of the UI. The object's position and path are displayed, as the objects motion and surroundings can assist in its identification.

The controls above the object panes allow the user to filter the objects displayed. False positives frequently result from very short detections, so setting a minimum frame length filters many of these out. The *by class* dropdown allows the user to choose to see previously classified objects for the purpose of reviewing them or unclassified objects for the purpose of annotating unlabelled objects. The *by file* dropdown allows the user to see only objects from a specific file. The buttons on the bottom right allow the user to assign a ground truth class to the currently selected object. Navigation and ground truth assignment can also be achieved through the use of keyboard shortcuts.

2) Hard negative mining and active learning: The object detection process yielded 53 million objects, 7.3 million of which were present for 15 frames or longer. Even with the filtering approaches listed above, searching the detections for rare objects among them can be time consuming. To make more optimal use of manual labeller's time we incorporated hard negative mining and active learning into the labelling tool.

Our implementation encourages a cyclic work-flow. The user can load predictions generated by a previously trained classifier into the labelling tool. The predictions can be used to prioritise the labelling of objects that will be most beneficial when training a new classifier. The new classifier is used to generate a new set of predictions that are loaded into the labelling tool for the next round of labelling.



Min length: 15 By class Jellyfish ☐ Sort by confidence

By file ALL 298 items

Brightness: ☐ Backg sub ☑ Show path ☐ Show point

Frame 57 / 181

Fig. 5. The labelling tool.

The labelling tool guides the users' choice of objects to label in two ways. The user may choose to view objects that have not yet been manually labelled, but have been predicted as belonging to a specified class. Incorrectly classified objects may stand out to the user, in which case they can provide the correct classification for these hard negatives.

The labelling tool can also sort objects in order of increasing prediction confidence. Predictions with low confidence correspond to objects that lie close to the decision surface and are more likely to be incorrectly classified. Labelling objects close to the decision surface results in larger gains in accuracy per additional label [6], in addition to making hard negatives easier to find.

3) Object identification data set: The labelling tool was used to identify 3,313 automatically detected objects. The composition of the resulting object identification data set is descibed in Table III. It was used to train our network as described below.

TABLE III
DATA SET USED TO TRAIN OBJECT CLASSIFIER

Class	Non-Oban	Oban	Total
Background	154	358	512
Sediment	0	189	189
Artefacts	187	812	999
Jellyfish	236	78	314
Fish	169	632	801
Seaweed	38	460	498
Total	784	2529	3313

E. Deep neural network classifier

1) Network architecture: Our deep neural network image classifier has an architecture inspired by the VGG-16 ImageNet [7] classifier of Simonyan *et al.* [8]. We also experimented with network architectures based on ResNets by He *et al.* [9], but found their performance to be slightly worse than that of our VGG style architecture. Our architecture is shown in Table IV.

TABLE IV
DEEP NEURAL NETWORK CLASSIFIER ARCHITECTURE

Description	Shape
48×48 greyscale image patch	$48 \times 48 \times 1$
Conv $3 \times 3 \times 32$, pad 1, batch norm	$48 \times 48 \times 32$
Conv $3 \times 3 \times 32$, pad 1, batch norm, stride 2	$24 \times 24 \times 32$
Conv $3 \times 3 \times 64$, pad 1, batch norm	$24 \times 24 \times 64$
Conv $3 \times 3 \times 64$, pad 1, batch norm, stride 2	$12 \times 12 \times 64$
Conv $3 \times 3 \times 128$, pad 1, batch norm	$12 \times 12 \times 128$
Conv $3 \times 3 \times 128$, pad 1, batch norm, stride 2	$6 \times 6 \times 128$
Conv $3 \times 3 \times 256$, pad 1, batch norm	$6 \times 6 \times 256$
Conv $3 \times 3 \times 256$, batch norm	$4 \times 4 \times 64$
Fully connected, 512 units	512
Fully connected, 6 units, softmax	6

2) Training: Our classifier was trained using the object identification data set described above. We used a learning rate of 1×10^{-4} with the Adam [10] optimisation algorithm. We trained our network for 200 epochs. The data set was split into 75%/12.5%/12.5% for training, validation and test. The per-patch error rate was evaluated after each epoch and used as a signal to drive early stopping [11]; the state of the network was saved for use after the epoch at which the lowest validation error rate was achieved.

At the start of training the mean and standard deviation of the patch pixel values of the training samples is computed in order to offset and scale them to have zero mean and unit variance. The data augmentation used during training consists of the following: random crops (58×58 pixel training patches are extracted from the sonar data so that random 48×48 crops can be extracted), random horizontal and vertical flips and X-Y swaps, random per-patch contrast modification by multiplying the pixel values by a value in the range $[0.4, 2.5]$ (logarithmically spaced) and random per patch brightness modification by offsetting the pixel values by a value in the range $[-3, 3]$.

V. Results

A. Object identification data set

The performance of our object classifier measured on the validation and test sets (as described above in section IV-E2) is summarised in the confusion matrices in Figures 6 and 7. The strong performance on the object identification data set is indicated by the strong diagonal line in both confusion matrices. The confusion between the background and sediment classes will have little impact on the effectiveness of our system as sediment detections will be ignored for the purpose of reporting jellyfish blooms. The sediment class was added for the purpose of completeness and to demonstrate the systems ability to quantify a variety of different types of object. The confusion between jellyfish and seaweed is of more concern. In situations where jellyfish are relatively rare, mis-detecting a proportion of instances of seaweed will results in a high false positive rate.

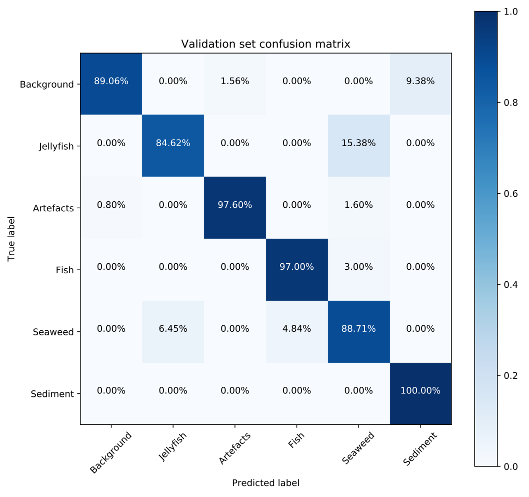


Fig. 6. Validation set confusion matrix

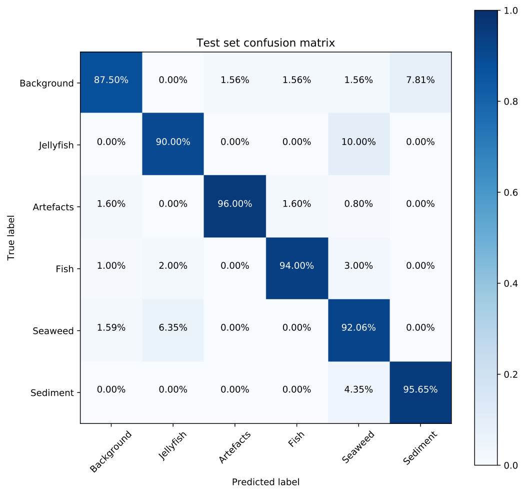


Fig. 7. Test set confusion matrix

B. Overall

Our object identification dataset was grown in four rounds using the cyclic work-flow described in section IV-D2. Each round consists of using predictions from the classifier trained in the previous round to guide manual labelling using active learning and hard negative mining to grow the data set, after which a new classifier is trained. The size of the object identification dataset in each round (identified by date) is presented in Table V and the predictions generated by a classifier trained in that round are presented in Table VI. It is worth noting that the class predictions include objects for which there are ground truth labels.

We consider the majority of jellyfish detections in Table VI to be false positives as there are only 78 known definite sightings. This number could increase if more are found with addition manual annotation effort.

TABLE V
OBJECT IDENTIFICATION DATA SET SIZE

Date	2017-11-17	2017-12-30	2018-02-07	2018-03-20
Class				
Background	478	479	494	512
Sediment	0	0	166	189
Artefacts	807	983	995	999
Jellyfish	239	273	293	314
Fish	517	666	743	801
Seaweed	0	235	309	498
TOTAL	2041	2636	3000	3313

TABLE VI
PREDICTIONS FOR DETECTED OBJECTS TRACKED FOR $>=7$ FRAMES

Date	2017-11-17	2017-12-30	2018-02-07	2018-03-20
Class				
Background	5738412	4965645	5068860	5474754
Sediment	0	0	0	3288
Artefacts	8003451	8461955	8367755	7978190
Jellfish	28643	4641	4421	3996
Fish	705961	982538	985809	882576
Seaweed	0	61688	49622	133663
TOTAL	14476467			

VI. CONCLUSIONS AND FUTURE WORK

The main limitation of the analysis of our results was due to the rarity of jellyfish presentations in the Oban 2017 data. Despite having captured 27 days of sonar footage during the Oban jellyfish season, only 78 definite sightings of individual jellyfish were found. This severely limits our ability to assess the effectiveness of our system in detecting jellyfish blooms as we have no blooms – in which a large number of individuals would be present – for our system to detect. The next stage of the JellyMonitor project involves further deployments of the prototype system, including a deployment adjacent to a coastal power station. The data gathered will permit a more rigorous analysis of the system and will hopefully contain numerous jellyfish sightings.

The classifier in our system uses only image patches as input. We would like to explore the use of motion paths generated by the object tracker as an additional input to the classifier.

During the project we noticed that artefacts that result from reflections from the surface of the water generate pattern with a repetitive appearance. Large numbers of artefact detections will tend to cluster around a region of the footage for a significant period of time. Detection of these clusters of detections could be used to suppress any false positives resulting from mis-identifying artefacts.

ACKNOWLEDGMENT

This project was jointly funded by Innovate UK (grant #102072) and EDF Energy. We would like to thank nVidia for the generous donation of an nVidia Jetson development kit.

REFERENCES

[1] C.-H. Han and S.-I. Uye, "Quantification of the abundance and distribution of the common jellyfish aurelia aurita s.l. with a dual-frequency identification sonar (didson)," *Journal of Plankton Research*, 2009.

[2] Sound Metrics Corp. Sound Metrics - ARIS Explorer 3000. [Online]. Available: http://www.soundmetrics.com/Products/ARIS-Sonars/ARIS-Explorer-3000

[3] NVidia Corp. Jetson TX2 Module - NVidia Developer. [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx2

[4] H. W. Sorenson, *Kalman filtering: theory and application*. IEEE, 1985.

[5] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, 1955.

[6] D. Wang and Y. Shang, "A new active labeling method for deep learning," in *IJCNN*, 2014.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[10] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[11] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.