2022

Naval Mine Detection and Seabed Segmentation in Sonar Images with Deep Learning

Luca Russo

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Naval Mine Detection and Seabed Segmentation in Sonar Images with Deep Learning

Luca Russo

This thesis is presented as part of the requirements for the conferral of the degree:

Master of Philosophy

Supervisor:
Prof. Son Lam Phung

Co-supervisors:
Prof. Christian Ritz

The University of Wollongong
School of Electrical, Computer and Telecommunications Engineering

September 2022

This research has been conducted with the support of University of Wollongong Research Scholarship.

Declaration

I, *Luca Russo*, declare that this thesis is submitted in partial fulfilment of the requirements for the conferral of the degree *Master of Philosophy*, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Luca Russo

March 5, 2023

Abstract

Underwater mines are a cost-effective method in asymmetric warfare, and are commonly used to block shipping lanes and restrict naval operations. Consequently, they threaten commercial and military vessels, disrupt humanitarian aids, and damage sea environments. There is a strong international interest in using sonars and AI for mine countermeasures and undersea surveillance. High-resolution imaging sonars are well-suited for detecting underwater mines and other targets. Compared to other sensors, sonars are more effective for undersea environments with low visibility.

This project aims to investigate deep learning algorithms for two important tasks in undersea surveillance: naval mine detection and seabed terrain segmentation. Our goal is to automatically classify the composition of the seabed and localise naval mines.

This research utilises the real sonar data provided by the Defence Science and Technology Group (DSTG). To conduct the experiments, we annotated 150 sonar images for semantic segmentation; the annotation is guided by experts from the DSTG. We also used 152 sonar images with mine detection annotations prepared by members of Centre for Signal and Information Processing at the University of Wollongong.

Our results show Faster-RCNN to achieve the highest performance in object detection. We evaluated transfer learning and data augmentation for object detection. Each method improved our detection models mAP by 11.9% and 16.9% and mAR by 17.8% and 21.1%, respectively. Furthermore, we developed a data

augmentation algorithm called Evolutionary Cut-Paste which yielded a 20.2% increase in performance. For segmentation, we found highly-tuned DeepLabV3 and U-Nett++ models perform best. We evaluate various configurations of optimisers, learning rate schedules and encoder networks for each model architecture. Additionally, model hyper-parameters are tuned prior to training using various tests. Finally, we apply Median Frequency Balancing to mitigate model bias towards frequently occurring classes. We favour DeepLabV3 due to its reliable detection of underrepresented classes as opposed to the accurate boundaries produced by U-Nett++. All of the models satisfied the constraint of real-time operation when running on an NVIDIA GTX 1070.

Acknowledgements

I'd like to thank the Defence Science and Technology Group for the opportunity to work on this project. A special thanks to Dr Weizhen Zhou and Dr Phillip Chappel for their discussions and feedback. Their domain knowledge and expertise has greatly helped this project.

Secondly, thank you to my fantastic supervisors, Prof. Son Lam Phung and Prof. Christian Ritz, for their constant support, advice and wisdom. I cannot stress the word constant enough. The commitment that my supervisors showed in helping me reach my goals is inspirational. Thank you Lam for many valuable lessons in approaching life with decisive actions, calmness and composure.

Finally, thank you to my family and friends for all the love and encouraging words over the past years.

Contents

Abstract iv

Acknowledgements vi

List of figures x

List of tables xv

Acronyms xv

1 Introduction 1

1.1 Research Objectives 3

1.2 Research Contributions 3

1.3 Thesis Organization 4

2 Literature Review 5

2.1 Overview of mine counter-measure operations 6

2.1.1 Sonar technology 6

2.1.2 Traditional mine-detection methods 8

2.2 Object Detection 10

2.2.1 Unified detection algorithms 12

2.2.2 Region-based detection algorithms 15

2.3 Semantic Segmentation 17

2.3.1 Encoder-Decoder networks 17

2.3.2 Multi-scale and pyramid networks . . . 19
2.3.3 Atrous convolution networks . . . 20

3 Naval Mine detection **23**
3.1 Detection dataset overview . . . 24
3.2 Proposed approach . . . 26
3.3 Evolutionary Data Synthesis . . . 29
3.3.1 Problem formulation . . . 29
3.3.2 Algorithm details . . . 30
3.4 Experiment details . . . 33
3.4.1 Introduction . . . 33
3.4.2 Dataset . . . 33
3.4.3 Evaluation metrics . . . 34
3.4.4 Experimental method . . . 35
3.4.5 Network training . . . 36
3.5 Results . . . 36
3.5.1 Transfer learning . . . 36
3.5.2 Data synthesis experiment . . . 39

4 Seabed segmentation **47**
4.1 Segmentation dataset overview . . . 48
4.2 Data annotation . . . 50
4.3 Proposed approach . . . 51
4.4 Experimental details . . . 56
4.4.1 Introduction . . . 56
4.4.2 Dataset . . . 57
4.4.3 Evaluation metrics . . . 57
4.4.4 Experimental method . . . 58
4.5 Results . . . 59
4.5.1 Learning rate range test . . . 59

4.5.2 Cyclic and OneCycle learninig rate schedules 65
4.5.3 Median frequency balancing 68
4.5.4 Comparing encoders 70
4.5.5 Comparing architectures 73

5 Conclusion **78**

5.1 Summary of work 78
5.2 Future research directions 79

List of Figures

2.1 A representation of a side-scan sonar sensor suite mounted on an autonomous underwater vehicle. The triangle originating from the sonar fish show the acoustic beams (white beams are emitted, yellow beams are reflected). 7

2.2 An example scenario where there is an object on either side of a sonar fish. From the figure, we can see how the shadow conveys information about the objects height. The distance of the object from the sonar fish also plays a role in shadow length, as objects that are further away will cast a longer shadow due to simple geometry. 8

2.3 YOLO architecture, as proposed in [39]. 13

2.4 SSD architecture, as proposed in [42]. 14

2.5 EfficientDet architecture, as proposed in [43]. 14

2.6 An overview of the R-CNN [47] detection model. 15

2.7 An overview of the Faster R-CNN [48] detection model. 16

2.8 FCN architecture, as proposed in [49]. 18

2.9 SegNet architecture, as proposed in [50]. 18

2.10 U-Net architecture, as proposed in [51]. 19

2.11 FPN architecture, as proposed in [53]. 20

2.12 The receptive field of a single feature in a 2-by-2 feature map. . . . 21

2.13 An example of the atrous algorithm in 1-D where kernel size = 3, input stride = 2 and output stride =1. 21
2.14 An illustration of the ASPP module [55]. 22
2.15 The DeepLabV3 model architecture [56]. 22

3.1 The Sea Scan® ARC Scout Mk II sonar sensor modules (black) and on-board processor unit (white). 24
3.2 The REMUS 100 AUV in a deployment scenario. In this image the vehicle is traveling to the right and out of the page. The black bar on the side of the vehicle is the sonar sensor. 25
3.3 Three labelled sonar images from the SMDD. Labelled objects are outlined in red. The difficulty of seafloor environment is presented in ascending order. (a) Smooth sand seafloor, (b) Ripple sand seafloor, and (c) Sand seafloor with dense rock formations. . 25
3.4 Three sonar snapshots from the SMDD. Each snapshot contains a labelled object at its center. These examples illustrate the variance of labelled MLOs. From left to right: (a) a semi-spherical MLO, (b) a cylindrical MLO parallel to sonar sensor and (c) a cylindrical MLO perpendicular to sonar sensor. 26
3.5 A sample sonar image (a) that has been: (b) horizontally flipped, (c) vertically flipped and (d) flipped both horizontally and vertically. The labelled object is outlined in red. The red arrows provide a frame of reference for the transformations 27
3.6 A synthetic sonar image with a 7×7 grid overlay. A MLO has been pasted in the center of the grid cell at column 5, row 5. The nadir region (row 4) is highlighted in red and is excluded. 30
3.7 (a) Single-point and (b) two-point crossover between pairs of selected individuals. In this example, each pair produces two children. 32

3.8 An example of mutation applied to a synthetic image. (a) Original synthetic image, (b) Synthetic image mutated with shifting, (c) Synthetic image mutated with replacement. 33

3.9 A visual example of the IoU calculation. In this example, the IoU score will be $\frac{1}{7}$ since the overlapping region is one-seventh the size of the total area made by the two boxes. 34

3.10 An image containing two objects (one person and one dog) with ground-truth labels and bounding boxes drawn (red and purple). In this example the detector model has made three predictions, shown in blue and numbered. Prediction 1 is a TP. Prediction 2 is a FP. Prediction 3 is a FN. 35

3.11 A comparison of mAP and frames per second of the models. Tiny YOLOv3 has the fastest inference time (46ms) and highest precision (21.43%). 38

3.12 A plot showing the mAR and mAP of the detection models with different training configurations. As expected, models trained from scratch with no data augmentation performed poorly. Both transfer learning and data augmentation lead to an improvement, however transfer learning had a greater benefit. Models trained with transfer learning and data augmentation performed on average 4 times better than the baseline. 39

3.13 Mean individual fitness across subsequent generations. The fitness score is based on validation loss (lower is better). We observe a clear downward trend over 40 generations. 41

3.14 Mean and median chromosome age across subsequent generations. The mean age of chromosomes in the population settled higher than the median age after initially increasing at the same rate. This is due to resilient chromosomes that last through many generations while new chromosomes are introduced to the gene pool. 42

3.15 Precision-Recall curve for the three models. 43
3.16 Two synthetic images (chromosomes) that survived for 37 generations. Bounding boxes are left loose so as to not occlude the object boundary. The objects blend well into the scene which makes for a realistic synthetic image. 44
3.17 Visual results of underwater mine detection produced by Tiny YOLOv3. *Top row*: Sonar images with manually annotated MLO ground-truths. *Bottom row*: Detection predictions of Tiny YOLOv3. 45
3.18 Further visual results of underwater mine detection produced by Tiny YOLOv3. *Top row*: Sonar images with manually annotated MLO ground-truths. *Bottom row*: Detection predictions of Tiny YOLOv3. 46

4.1 Three annotated images from the SBSD. 49
4.2 LabelMe UI displaying an annotated sonar image. 50
4.3 The OneCycleLR scheduler. 52
4.4 An example of exploding gradients. 53
4.5 Baseline results for the LRRT. 59
4.6 Weight decay LRRT for densenet121 61
4.7 Weight decay LRRT for efficientnet-b0. 62
4.8 Weight decay LRRT for mobilenet_v2. 62
4.9 Weight decay LRRT for vgg13_bn. 63
4.10 Weight decay LRRT for rensenet18. 64
4.11 Weight decay LRRT for dpn68. 64
4.12 Optimiser and learning rate scheduler comparison: Accuracy. . . . 66
4.13 Optimiser and learning rate scheduler comparison: Mean accuracy. 66
4.14 Optimiser and learning rate scheduler comparison: Mean IoU. . . . 67
4.15 Optimiser and learning rate scheduler comparison: F1 score. 67
4.16 Median frequency balancing - accuracy. 69

4.17 Median frequency balancing - mean accuracy. 69
4.18 Median frequency balancing - F1 score. 70
4.19 Median frequency balancing - mIoU. 70
4.20 Encoder test mean accuracy. 71
4.21 Encoder test F1 scores. 72
4.22 Encoder test mean IoU. 72
4.23 Architecture test mean accuracy. 74
4.24 Architecture test F1 scores. 75
4.25 Architecture test mean IoU. 75
4.26 A visual comparison of the outputs from the two best segmentation models. 76
4.27 A visual comparison of the outputs from the two best segmentation models cont. 77

List of Tables

2.1 Common preprocessing and detection stage techniques. 9

2.2 A summary of popular object detection benchmark datasets. . . . 11

3.1 An overview of the SMDD. 25

3.2 Evolutionary algorithm terms defined in the context of data augmentation. 32

3.3 The mAP @ 0.5 IoU scores of the detection models 37

3.4 The mAR scores of the detection models 37

4.1 An overview of the SBSD. 48

4.2 Class distribution in the SSD. 55

4.3 Baseline results for the LRRT. 60

4.4 Optimal learning rate and weight decay for each model. 65

4.5 The results of the encoder test. 71

4.6 The results of the architecture comparison experiment. 73

Acronyms

API Application Programming Interface.

AR Average Recall.

ASPP Atrous Spatial Pyramid Pooling.

AUV Autonomous Underwater Vehicle.

CNN Convolutional Neural Network.

CRF Conditional Random Field.

DL Deep learning.

DNN Deep Neural Network.

DSTG Defence Science and Technology Group.

ECP Evolutionary Cut and Paste.

FCN Fully Convolutional Network.

FN False Negative.

FP False Positive.

FPN Feature Pyramid Network.

GUI Graphical User Interface.

IoU Intersection over Union.

K-NN K-Nearest Neighbours.

LRRT Learning Rate Range Test.

mAP Mean Average Precision.

mAR Mean Average Recall.

MCM Mine Counter-Measure.

MFB Median Frequency Balancing.

ML Machine Learning.

MLO Mine-Like Object.

MRF Markov Random Field.

NMS Non Maximum Suppression.

ODF Ordinary Differential Function.

R-CNN Region-based Convolutional Neural Network.

RF Receptive Field.

RoI Region of Interest.

RPN Region Proposal Network.

SBSD Seabed Segmentation Dataset.

SGD Stochastic Gradient Descent.

SMDD Sonar Mine Detection Dataset.

SNR Signal-to-Noise Ratio.

SOTA State Of The Art.

SSD Single-Shot Detector.

SVM Support Vector Machine.

TL Transfer Learning.

TN True Negative.

VGG Visual Geometry Group.

WD Weight Decay.

YOLO You Only Look Once.

Chapter 1

Introduction

Chapter contents

1.1 Research Objectives 3
1.2 Research Contributions 3
1.3 Thesis Organization 4

Surveying underwater regions is a crucial task in a wide range of defence and industrial applications. Seabeds are composed of many distinctive materials and are home to flora, fauna and man-made devices. The composition of seabeds is of special interest to defence and infrastructure operations due the build up of hazardous devices and weaponry from past conflicts.

Naval mines are an effective means of controlling waterways that provide significant strategic and commercial benefits. From 1950 to 1998, naval mines caused more casualties to U.S Navy personnel than missiles, torpedoes, and aerial attacks combined [1]. Mines are an increasing threat in the world, with the number of mine-producing countries increasing by 75% between 1988 and 2000 [2]. Additionally, there are WW1 era mines that are still capable of causing substantial damage to modern warships. For example, the USS Tripoli and Princeton together sustained $125 million in damage after colliding with vintage mines off the coast of Iraq during the Gulf War [3].

In recent years, sonar-equipped autonomous underwater vehicles (AUVs) have made it possible to survey seafloor regions automatically. This has greatly improved the rate at which sonar seafloor data is obtained. As a result, the main bottleneck for MCM operations is now the number of trained operators that are available for data analysis. Currently, operators must manually examine a sonar image feed to detect mine-like objects (MLOs) in real time. This task is extremely time consuming. Additionally, MLOs rarely occur which causes boredom and fatigue [4]. Integrating deep learning models into advanced robotics systems can completely automate the search phase of MCM operations: saving time, money, and lives.

Deep learning models are very large artificial neural networks. They learn to make predictions by tuning a non-linear function that maps inputs to their desired outputs. This process, called training, generally requires that the network observers millions of input-output pairs. Models are prone to memorising inputs when given too few data points. This phenomenon is known as overfitting. It is characterised by poor model performance on unseen data. Furthermore, a key assumption of machine learning is that unseen data comes from the same distribution as training data. Both of these issues arise when considering sonar data.

Collecting sonar data involves laying mines, deploying a sonar-equipped AUV, and image labelling by a specialist. The result is small sonar datasets (hundreds or thousands of samples) that contain very few unique MLOs and seafloor environments. Conversely, real world MCM operations deal with highly variable mine types and seafloor environments [2]. This means that the already small datasets are not representative of possible deployment scenarios. Furthermore, environmental factors, such as temperature, salinity, water clarity and bathymetry can severely impact sensor performance [5, 6], which skews the data distribution of otherwise comparable scenarios.

Several techniques have been used to overcome sonar data limitations. Transfer learning makes use of large datasets to train CNNs that are then fine-tuned for underwater mine detection [7–11] or used as feature extractors for specialised classifiers [7, 10]. Data augmentation, which involves applying image processing transformations to increase the size of the training dataset, has been successfully applied to underwater mine detection in multiple works [10, 12]. Compact CNNs have been found to reduce network overfitting by lowering the number of tunable weights [12, 13].

1.1 Research Objectives

The aim of this research project is to investigate, implement and evaluate current deep learning algorithms to detect naval mines and classify seabed regions in real-time. The project combines the sonar imaging technology, object detection and image segmentation to create robust computer vision models.

1.2 Research Contributions

The contributions of the thesis can be highlighted as follows:

- We combine sonar imaging with object detection and semantic image segmentation algorithms for real-time naval mine detection and sea floor classification.
- We propose a novel data synthesis algorithm, called evolutionary cut-paste, as a way of generating additional data samples from our training data.
- We annotate a set of 156 sonar images to prepare them for training a semantic segmentation model.
- We evaluate SOTA object detection and semantic segmentation methods and evaluate their performance with our sonar dataset.

1.3 Thesis Organization

The thesis is structured as follows:

- **Chapter 1** introduces our research approach and objectives, and outlines our major contributions.
- **Chapter 2** highlights the importance of our research problem and gives a review of relevant literature.
- **Chapter 3** describes our dataset, experimental method and results for sonar mine detection.
- **Chapter 4** describes our dataset, experimental method and results for seabed semantic segmentation.
- **Chapter 5** summarises our research findings and proposes future research directions.

Chapter 2

Literature Review

Chapter contents

2.1 Overview of mine counter-measure operations 6

2.1.1 Sonar technology 6

2.1.2 Traditional mine-detection methods 8

2.2 Object Detection 10

2.2.1 Unified detection algorithms 12

2.2.2 Region-based detection algorithms 15

2.3 Semantic Segmentation 17

2.3.1 Encoder-Decoder networks 17

2.3.2 Multi-scale and pyramid networks 19

2.3.3 Atrous convolution networks 20

This section provides an overview of the literature from fields related to underwater surveillance and deep learning. Section 2.1 gives an overview of mine-countermeasure operations and sonar technology. Section 2.2 focuses on object detection methods. Section 2.3 explores deep learning architectures for image segmentation.

2.1 Overview of mine counter-measure operations

2.1.1 Sonar technology

Active sonar is a remote sensing technique that uses acoustic waves to observe underwater regions. The word sonar is an abbreviation for sound navigation and ranging [14]. Since the development of the first prototype echo sounder [15] in 1935, acoustic waves have been the dominant medium for underwater imaging. The main reason being that acoustic waves travel through water more effectively than electromagnetic waves [16].

The fundamental principle in sonar is reflection. Similarly to echolocation in bats [17] and dolphins [18], sonar works by emitting a pulse of sound waves and measuring the difference in the transmitted and received wave forms. In the case of side scan sonar, a side-mounted ceramic transducer is excited by an electronic signal to produce a monochromatic or chirp pulse. The pulse then propagates into the environment where it is reflected back to a receiving transducer that converts the signal back into an electronic waveform, as shown in Figure 2.1. The sonar equation [6] characterises how sonar images are produced considering all elements of the transmission. Equation 2.1 describes how much energy must be returned to the transducer for a detection to occur.

$$DT \leq SL - 2 \times TL + TS - (NL - DI) \tag{2.1}$$

The detection threshold (DT) represents the minimum return signal strength for a reflection to be registered and the source level (SL) is the strength of the transmitted wave. The remaining terms describe the transmission of the acoustic wave - from source to target and back. The transmission loss (TL) is the loss of the sound wave as it propagates through water, target strength (TS) is the amount of sound reflected back by the target, noise level (NL) is a measure of background noise at the transmitter and directivity index (DI) is a measure of the concentration of a signal in a certain direction.

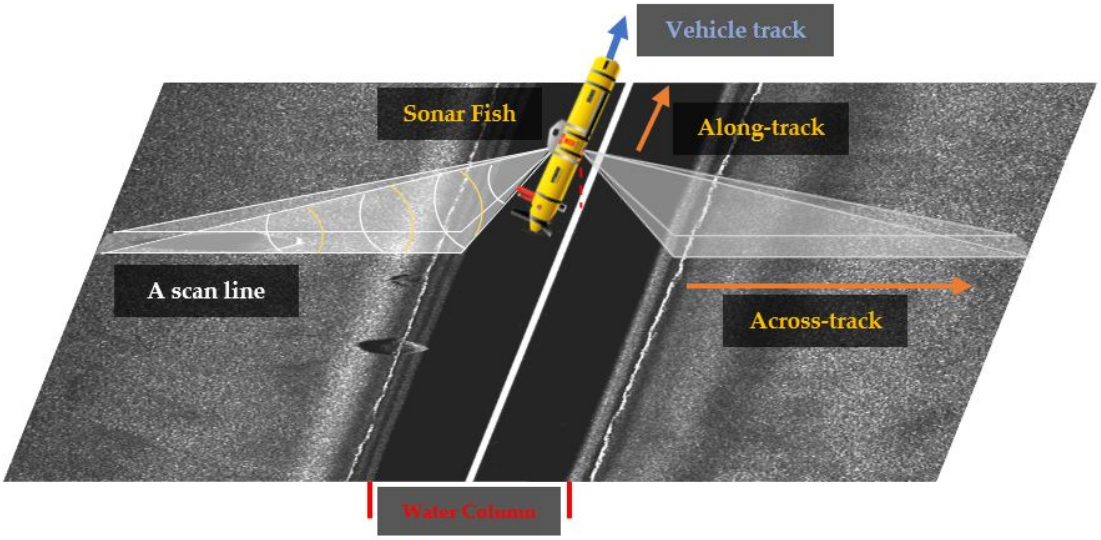


Figure 2.1: A representation of a side-scan sonar sensor suite mounted on an autonomous underwater vehicle. The triangle originating from the sonar fish show the acoustic beams (white beams are emitted, yellow beams are reflected).

A functional understanding of sonar technology is useful to interpret sonar images correctly. Each object in a sonar image is made up of a highlight region and a shadow. The highlight is the portion of the object that reflects acoustic waves back to the sonar fish. The intensity of pixels in a highlight are determined by target strength, which itself is governed by the object's reflectivity. Reflectivity is based on the target object's size, shape, composition and thickness as well as the sonar's frequency, pulse duration and angle of incidence with the object [16]. Shadows are highly descriptive characteristics of sonar images due to the geometry of the sidescan sonar imaging method. Figure 2.2 shows that the height of an object on the seafloor can be deduced from the length of the shadow relative to the distance from the tow-fish.

The quality of sonar images are highly dependent on environmental conditions. The noise level and transmission loss from Equation 2.1 are the major factors that determine sonar image quality. Both variables affect image contrast and the noise level introduces additional image artifacts. Contrast is measured as the signal-to-noise (SNR) ratio of the system. It is the ratio of the intensity of an object highlight against its shadow.

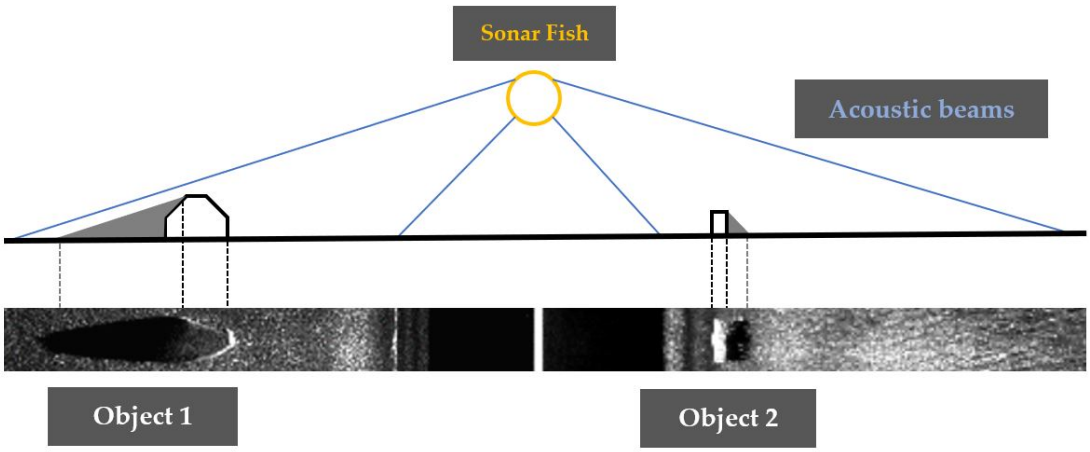


Figure 2.2: An example scenario where there is an object on either side of a sonar fish. From the figure, we can see how the shadow conveys information about the objects height. The distance of the object from the sonar fish also plays a role in shadow length, as objects that are further away will cast a longer shadow due to simple geometry.

2.1.2 Traditional mine-detection methods

Historically, mine detection algorithms mainly employed traditional image processing techniques from the 1980s to late 2000s. These early systems divide mine detection into four interconnected processing tasks: (1) preprocessing, (2) detection, (3) feature extraction, and (4) classification. Table 2.1 shows common preprocessing and detection techniques that are found across many works. This review focuses on novel detection, feature extraction and classification stages of traditional mine detection systems.

Russell and Lane [27] were among the first to propose an automated mine detection. They developed a knowledge-based system that uses hierarchical data representations of the environment and the vessel's state to automatically generate an operating rule-set. The system is based around a globally accessible memory unit called the blackboard that could be written to by rule-generating modules, called knowledge sources (KS). At each time step, the KS modules can be triggered to create an agenda for updating the blackboard. The authors state that the feedback structure of the blackboard cell framework demonstrated basic AI functionality by automatically inferring expert knowledge. However, the system

	Approach	Representative Works	Description
Preprocessing	Median Filter	*Ciany & Huang* [19] *Johnson & Deaett* [4] *Dobeck et al.* [20] *Lane & Stoner* [21]	Remove speckle and random noise by averaging neighbouring pixels.
	Image Normalisation - Range normalisation - Histogram equilisation	 *Dobeck et al.* [20] *Chang et al.* [22]	Improve the contrast between object highlights and shadows
Detection	Thesholding - Fixed amplitude - Adaptive amplitude - Pixel count	 *Johnson & Deaett* [4] *Dobeck et al.* [20] *Ciany & Huang* [19] *Dobeck et al.* [20]	Identify important image regions using simple inequalities to group pixels based on intensity. Additionally, the size of groupings are used to eliminate insignificant regions.
	Matched Filter	*Dobeck et al.* [20] *Williams et al.* [23] *Pinto et al.* [24]	Compare an image region to templates to determine whether the region is likely to contain a similar object.
	Anomaly Detector	*Nevis et al.*[25] *Strand* [26]	Identify rare objects by identifying statistical outliers.

Table 2.1: Common preprocessing and detection stage techniques.

was far ahead of its time and lacked proper testing to substantiate the claim.

Johnson and Deaett [4] proposed a toxic-waste deposit recogniser that uses three manually selected features and rules based on novice-operator knowledge. The aim was to identify regions of interest for further analysis by an expert operator. Three features were established through conversations with operators who outlined the decision making process. The features were: (1) first and second geometric moments, (2) the shape factor (a ration of perimeter to area), and (3) the distinctiveness (the SNR of a particular region). These features are passed into a function to produce a composite score, which, if greater than a predetermined threshold, is marked for expert analysis.

Dobeck et al. [20] developed a multi-classifier system that selects features from a shared feature pool, called the AMDAC. The AMDAC uses a stepwise feature selection strategy to evaluate subsets of features for two unique classifiers in parallel. Each step involves evaluating the performance of a classifier, then adding

or removing features to the subset until no improvements occur. This strategy combines the outputs of two classifiers (K-NN and ODF) that have complimentary features due to the difference in mathematical formulation. The rate of false positives was reduced by a factor of four by using two classifiers in parallel.

Reed et al. [28] proposed a novel co-operating statistical snake (CSS) model for extracting object highlights and shadow regions. Prior information regarding typical mine shapes and geometry were used for a detection-orientated unsupervised MRF to segment the image into regions of shadow, seafloor reverb and object highlights. This method proved successful in more complex scenes (sand ripples). The authors then designed a classification model [29] based on Dempster-Shafer information theory using the improvement from [28].

Williams et al. [30] identified data imbalance as a major challenge for underwater mine detection. They proposed labelling clutter objects as a way to reduce the number of false detections. Mines were made easier to distinguish by training a classifier to recognise the features of clutter objects, such as rocks. Two matched filters detect regions that are likely to contain targets, then two additional filter kernels calculate statistical qualities of the regions. The authors found that targets were generally well represented by their mean values.

2.2 Object Detection

The goal of generic object detection algorithms is to localise and classify all object instances for known classes that are present in a given image. Detection is a preliminary task for systems performing complex reasoning tasks in the real-world. As such, natural images are the main focus of generic object detection algorithms since they are applicable to a wide range of problem domains and contain many common object classes. In the past decade, deep convolutional neural networks have been the leading solution for object detection due to their ability to learn object representations across many levels of abstraction. [31].

Current SOTA deep learning algorithms need millions of training samples to learn robust object representations that are invariant to scale, rotation, orientation and context. Open-source benchmark datasets, such as Pascal VOC [32], MS COCO [33], ImageNet[34] and Open Images[35], have driven progress in generic object detection. A summary of popular benchmark datasets is given in Table 2.2. The benefits of large open-sourced datasets is three-fold. First, they provide the data needed to train large deep learning models. Secondly, they standardise the evaluation of competing SOTA models. Finally, new datasets progress the field of object detection by introducing increasingly challenging scenes.

Table 2.2: A summary of popular object detection benchmark datasets.

Dataset name	Year Started	Total images	Categories	Description
PASCAL VOC [32]	2005	11,540	20	The first standard object detection used as a benchmark in research. Images often contain multiple objects. Objects may be obscured or occluded.
ImageNet [34]	2009	~ 14 million	21,841	A significant increase in categories and total images. Images are centered around objects and are less object-dense.
MS COCO [33]	2014	~ 330,000	91	Images are very close to real-world scenarios with a high density of objects per image. Images are frequently clustered and overlap.
Open Images [35]	2017	~ 9 million	~ 500	Semi-automatically annotated dataset with only 2 million human-annotated images. Object density is similar to MS COCO. The only dataset to have variable image size.

Modern deep learning approaches to object detection can be divided into two categories: unified detection methods and region-based detection methods. The key difference between the categories is that unified algorithms transform inputs image directly to output bounding boxes and classes with regression, whereas region-based systems require a separate method for generating region proposals. In the following sections we will cover the major contributions to both single-stage and two-stage object detection algorithms.

2.2.1 Unified detection algorithms

Unified detection algorithms frame object detection as a regression problem from image pixels directly to class predictions and bounding box coordinates. Detection involves a single forward pass through the network which allows much faster inference times than two-stage algorithms. Additionally, single-stage networks can be optimised end-to-end during training which leads to easier implementation and optimisation.

Szegedy et al. [36] were the first to formulate object detection as a regression task. Their proposed D-CNN, DetectorNet, replaces the softmax classification layer in AlexNet [37] with a regression layer. DetectorNet uses five networks in total. The first network generates a coarse binary mask indicating foreground regions. The binary mask is used to produce image crops. Each crop is passed through the four remaining networks to predict masks for the objects' top, bottom, left and right halves. Finally, the four masks are converted to a single bounding box through simple inference. Using multiple image crops results in slower inference but reduces the uncertainty of the model.

Sermanet et al. [38] proposed OverFeat, the first single-stage object detector based on fully convolutional networks (FCNs). OverFeat introduced the idea of using 1×1 sized convolutional kernels to replace the fully connected layers that were standard for regression. This change allows the use of multiscale input images because FCNs are not limited to fixed-size inputs to ensure correct feature map dimensions. Predictions across multiple scales are combined using a greedy merge strategy to improve performance. OverFeat was a significant leap forward in single-stage detectors, winning ILSVRC2013 and directly inspiring newer single-stage detectors such as YOLO and SSD.

Redmon et al. [39] proposed YOLO (You Only Look Once), a novel framework that detects objects with a single forward pass. Unlike OverFeat, YOLO only needs to see one full-sized image without upsampling. YOLO divides the input image

into an S × S grid. B bounding boxes, C class probabilities and a confidence score for each box are predicted within each grid cell. The output predictions are filtered by a confidence score threshold, and duplicate detections are eliminated with non-maximal suppression (NMS). It is important to note that the size of the grid is dependent on the input resolution. The original paper used 448 × 448 images that are down-scaled to a 7 × 7 grid through strided convolution operations. The coarseness of the grid leads to poor detection of small objects that only take up a fraction of a grid cell. YOLO's unified framework led to faster training and inference but was less accurate than two-stage detectors.

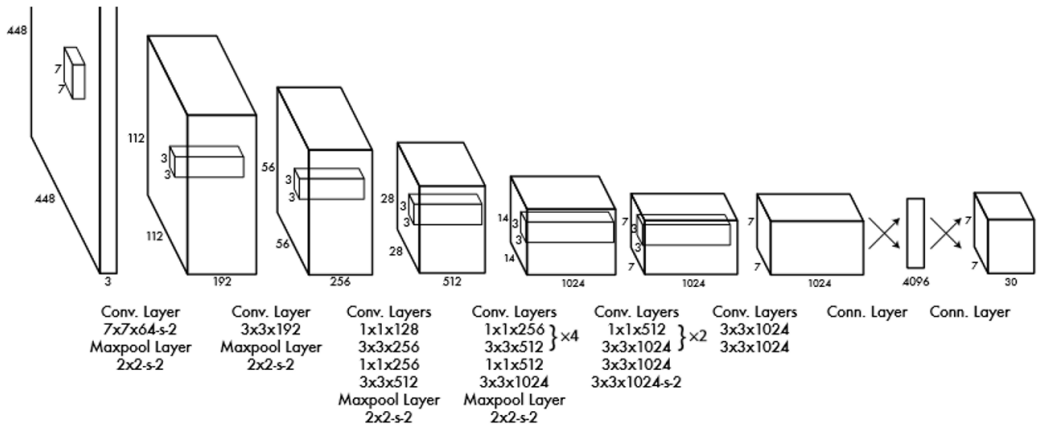


Figure 2.3: YOLO architecture, as proposed in [39].

Redmon and Farhadi [40] advanced the YOLO architecture with YOLOv2. Most notably, batch normalisation, multiscale training and anchor boxes generated with k-means greatly improved accuracy. The computational cost of these improvements were negated by replacing GoogLeNet with a smaller network, DarkNet19. Finally, Redmon and Farhadi [41] address the shortcoming of the YOLO architecture in detecting small objects by adding multi-scale predictions in the new DarkNet53-based YOLOv3 architecture.

Liu et al. [42] proposed SSD, a single shot multibox detector that builds upon OverFeat and YOLO by performing detections at multiple object scales from feature maps of varying sizes. SSD adds additional convolutional layers to a

pre-trained VGG16 backbone feature extractor network. Feature maps produced by both the backbone VGG16 network and added convolutional layers are used to make predictions. The network predicts offsets to a fixed number of default boxes for each cell of the various feature maps. Similarly to YOLO, SSD uses non-maximum suppression to remove duplicate predictions.

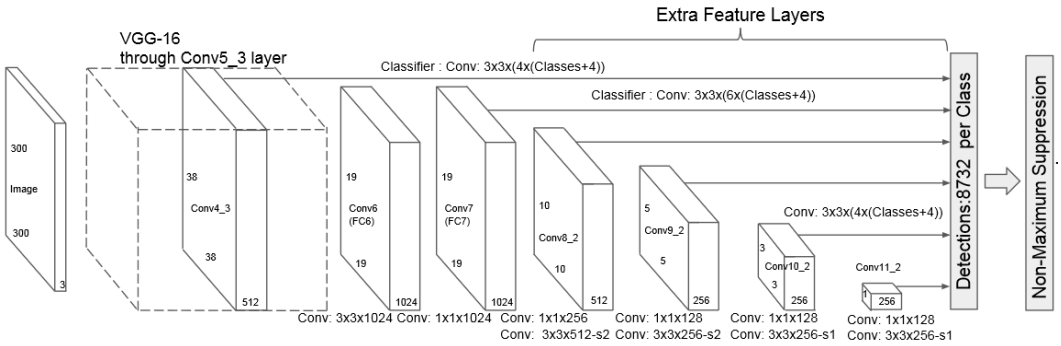


Figure 2.4: SSD architecture, as proposed in [42].

Tan et al. [43] proposed EfficientDet, a combination of recent AutoML learned feature extractor networks, called EfficientNets [44], and a novel bi-directional feature pyramid network (BiFPN). The EfficientDet architecture employs a uniform scaling coefficient to adjust the overall network shape. The coefficient, ϕ, scales the backbone network width and depth, the BiFPN depth and number of channels, the box and class prediction networks, and the input image resolution. The result is a flexible, fast and accurate single stage detector that fuses features at five feature map scales.

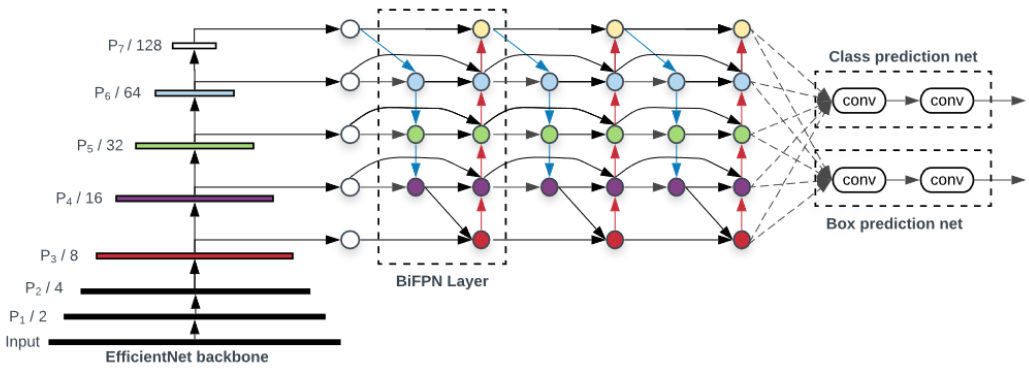


Figure 2.5: EfficientDet architecture, as proposed in [43].

2.2.2 Region-based detection algorithms

Region-based algorithms are characterised by the additional region proposal operation for generating candidate regions. Early algorithms use traditional image processing techniques to generate class-agnostic region proposals. Selective search [45] generates thousands of region proposals by over-segmenting images based on colour, texture, shape and size then iteratively grouping similar segments. More recently, selective search was replaced by region proposal networks (RPNs) [46], which are FCNs specifically trained to generate region proposals. Historically, two-stage detectors have achieved higher accuracy across benchmark detection, however they are slower due to the quantity of image region crops that need to be processed.

Girshick et al. [47] were the first to merge CNNs with a region proposal network to develop the region-based object detection framework, R-CNN. R-CNN uses selective search [45] to generate regions which are then warped into a standard size and passed into a CNN model for feature extraction. The features are used to train a bounding box regressor network and class specific SVM classifiers for each object category. The multistage pipeline achieves very good levels of recall and precision but is slow and difficult to train due to its disparate stages. Furthermore, R-CNN is memory intensive since features need to be extracted for each region proposal.

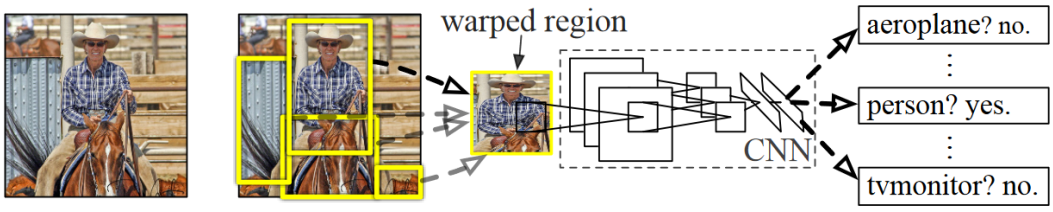


Figure 2.6: An overview of the R-CNN [47] detection model.

The speed and detection of the R-CNN family of models was improved in Fast R-CNN [46]. The main contribution of Fast R-CNN was the sharing convolutions across region proposals and the addition of a Region of Interest (RoI) pooling layer

before the fully connected layer. RoI pooling minimises data loss due to warping by approximating the distortions. Faster R-CNN [48], the next iteration of the R-CNN algorithm, replaces the selective search algorithm for region proposal with a separate RPN. The RPN uses the same backbone network to perform region proposals by initialising and tuning k anchor boxes with regression.

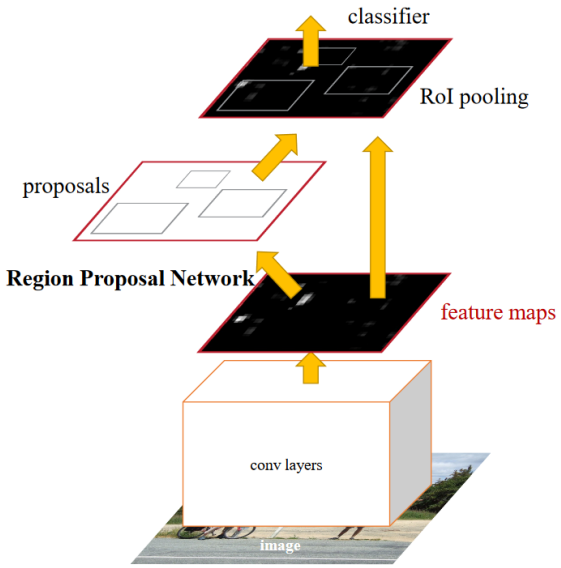


Figure 2.7: An overview of the Faster R-CNN [48] detection model.

2.3 Semantic Segmentation

Semantic segmentation is a fundamental task in computer vision applications such as autonomous driving, medical image analysis, surveillance and augmented reality. Semantic segmentation algorithms generate pixel-wise masks that assign each pixel in the input image with a value corresponding to the class the pixel belongs to. Recently, deep learning methods have dominated semantic segmentation tasks, vastly outperforming traditional image processing techniques.

In this chapter, we discuss the current SOTA approaches to semantic segmentation using deep learning. We will cover encoder-decoder, feature pyramid, attention-based and atrous convolution networks.

2.3.1 Encoder-Decoder networks

Encoder-decoder networks are made up of two components. As the name suggests, they are the encoder and decoder. Fundamentally, an encoder-decoder network compresses an input image into a low-dimensional representation, called a feature map, which is then up-scaled into a segmentation mask. The encoder is responsible for producing the feature map. Input images are compressed by applying consecutive convolution and pooling operations. Conversely, the decoder takes the feature map as input and up-scales it through deconvolution operations. The output of the decoder is a segmentation mask with equal resolution to the input image.

Long et al. [49] were the first to apply an encoder-decoder architecture to semantic segmentation. Their FCN is composed of convolutional and pooling layers, and can be trained end-to end. FCN performs upsampling in a single step, unlike modern encoder-decoder networks. Upsampling is performed by learned deconvolutional kernels. The final feature map is fused with high resolution feature maps from previous layers to give a fine-grain segmentation map.

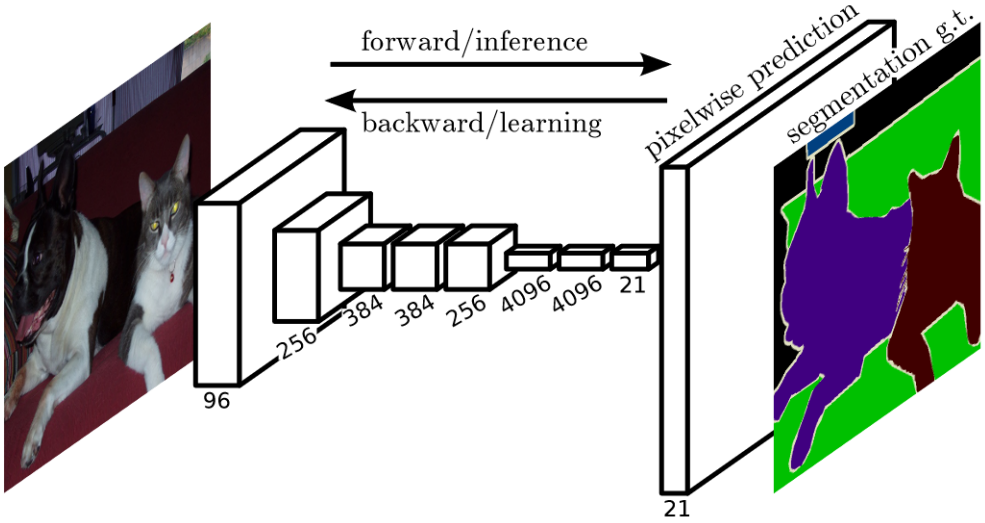


Figure 2.8: FCN architecture, as proposed in [49].

Badrinarayanan et al. [50] proposed a new encoder-decoder architecture called SegNet. The main contribution of SegNet is a deeper decoder network which mirrors the encoder. For each pair of convolution and pooling operations in the encoder, there is a corresponding deconvolution operation in the decoder. Pooling indicies from the encoder are passed to the decoder to perform non-linear upsampling. The upsampling layer contains sparse feature maps which are made dense through convolution in subsequent layers. This removes the need for learned deconvolution kernels which increase the memory requirements of the network.

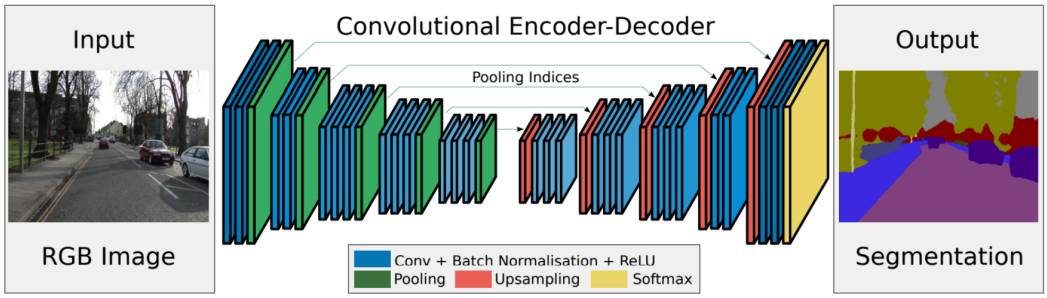


Figure 2.9: SegNet architecture, as proposed in [50].

Ronneberger et al. [51] proposed U-Net, an encoder-decoder network for medical image analysis. U-Net follows the mirrored encoder-decoder architecture

introduced by SegNet. The main difference between the two architectures is that U-Net passes feature maps rather than pooling indices between the encoder and decoder part. Feature maps are passed directly between the layers with skip connections. Additionally, U-Net uses a large number of channels in the decoder part which conserves contextual information throughout the upscaling process.

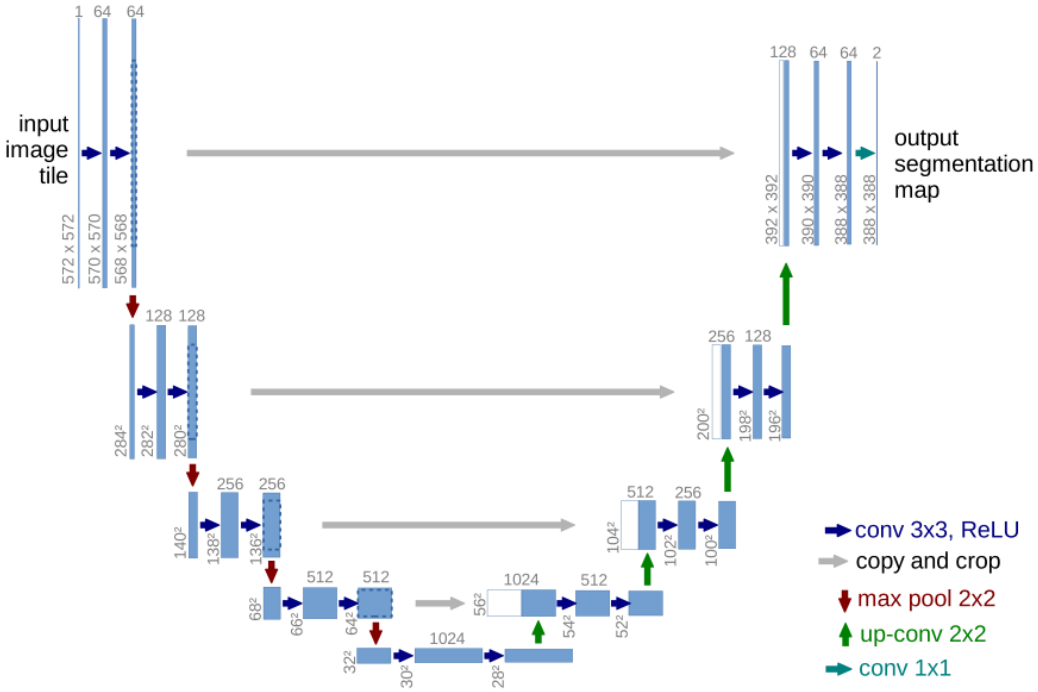


Figure 2.10: U-Net architecture, as proposed in [51].

2.3.2 Multi-scale and pyramid networks

Deep learning models often struggle to segment small objects accurately. Multi-scale and pyramid networks aim to improve the performance of segmentation models by making predictions at multiple image scales and combining the predictions.

Lin et al. [52] proposed the feature pyramid network (FPN) for object detection. The architecture was later extended by Kirillov et al. [53] for image segmentation. FPN uses strided convolutions to extract features while down scaling the feature map. Contextual information is preserved through the use of increasing channels throughout the feature extraction process. The high di-

mensional feature maps are projected to a lower dimensional space with 1-by-1 convolution. Feature maps in the top-down pathway are upscaled using nearest neighbour interpolation and combined to the projections from the bottom-up pathway. Finally, the feature maps from each scale are assembled to produce the final segmentation mask.

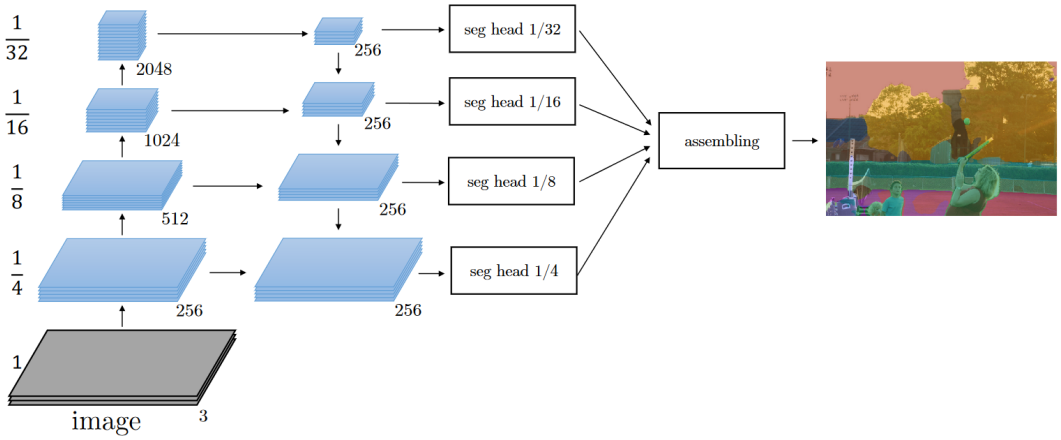


Figure 2.11: FPN architecture, as proposed in [53].

2.3.3 Atrous convolution networks

Even for humans, interpreting images accurately requires a spread focus over the entire content of the image. It is easy to incorrectly classify a surface if we are zoomed in too far on the region. For a deep learning model, this is analogous to having a feature with a small receptive field (RF). The RF describes how many pixels in the input image have had an effect on a given feature of the feature map. Convolutions are applied across subsequent layers in deep networks such that a single feature in a feature map can be influenced by a large number on pixels in the input image. Figure 2.12 shows a simple example of the receptive field.

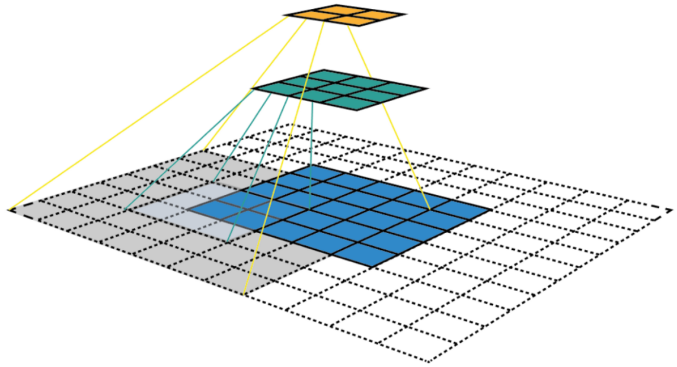


Figure 2.12: The receptive field of a single feature in a 2-by-2 feature map.

Atrous convolution networks were created as a way to achieve a larger receptive field without increasing computation or memory requirements. First proposed by Chen et al. [54], DeepLab has been iterated three times [55–57] and continues to be a prevalent model in segmentation tasks. DeepLabV1 [54] aimed to solve the issue of poor localisation in DNN segmentation models by combining the output of the final layer of their model with a long-range conditional random field (CDF) [58]. Up-scaling is applied to the output segmentation mask by means of bilinear interpolation between the fields nodes. In this iteration of DeepLab, atrous convolutions are applied to 1-D vectorised representations of the images rather than by 2-D convolution kernels used in later DeepLab models. Figure 2.13 shows the atrous algorithm implemented in [54].

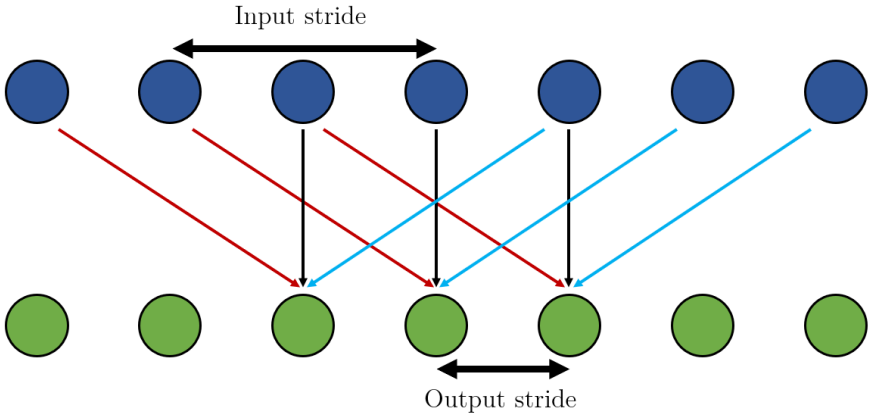


Figure 2.13: An example of the atrous algorithm in 1-D where kernel size = 3, input stride = 2 and output stride =1.

DeepLabV2 [55] extended the idea of atrous convolution into the 2-D plane with the introduction of the atrous spacial pyramid pooling (ASPP) module. The ASPP works by fusing feature maps with varied levels of contextual information contained in each. Atrous convolutions are a means to control the size of the receptive field that a feature map has. As such, the amount of the image that is seen by each feature map can be predetermined, as we see in Figure 2.15.

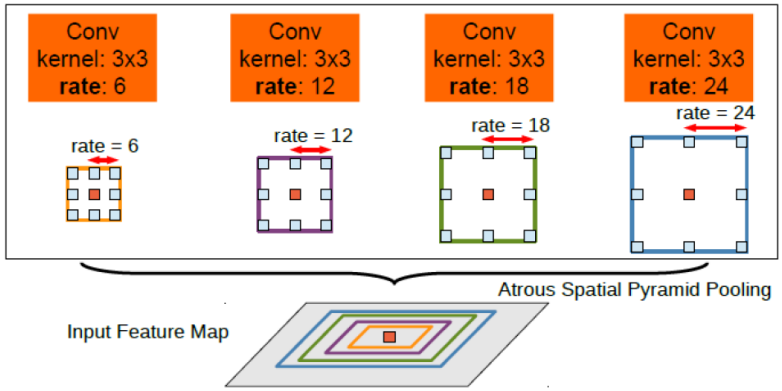


Figure 2.14: An illustration of the ASPP module [55].

The DeepLab model is further iterated on by Chen et al. with the introduction of DeepLabV3 [56] with two main contributions, which are multi-scale context and the removal of CRFs from the final layer of the network. In DeepLabV3, the authors also propose the implementation of atrous convolutions in cascade and parallel configurations for more fine-grained control of receptive fields. CRFs are replaced by image-level features that provide global context and improve localisation while also boosting performance.

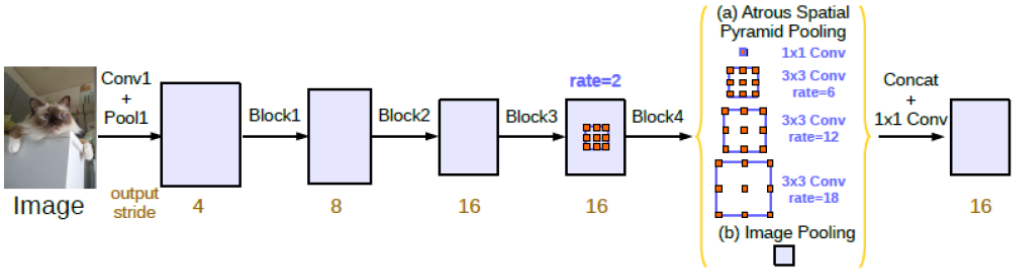


Figure 2.15: The DeepLabV3 model architecture [56].

Chapter 3

Naval Mine detection

Chapter contents

3.1 Detection dataset overview 24
3.2 Proposed approach 26
3.3 Evolutionary Data Synthesis 29
3.3.1 Problem formulation 29
3.3.2 Algorithm details 30
3.4 Experiment details 33
3.4.1 Introduction 33
3.4.2 Dataset 33
3.4.3 Evaluation metrics 34
3.4.4 Experimental method 35
3.4.5 Network training 36
3.5 Results 36
3.5.1 Transfer learning 36
3.5.2 Data synthesis experiment 39

Our first objective is to evaluate existing deep learning algorithms in underwater mine detection. Section 3.1 provides an overview of the sonar detection dataset. Section 3.2 outlines the proposed approaches to applying deep learning. Section 3.3 details our proposed data synthesis algorithm. Section 3.4 describes the experimental methods. Finally, Section 3.5 presents the results of our experiment.

3.1 Detection dataset overview

Our research uses the Sonar Mine Detection Dataset (SMDD) provided by the Defence Science and Technology Group. The SMDD was compiled during a naval mineshape recovery operation in Jervis Bay, NSW, Australia. The sonar data was captured using a Sea Scan® ARC Scout Mk II side scan sonar (Figure 3.1) with dual frequency channels operating at 900kHz and 1800kHz. The 900kHz channel has 0.2m vertical resolution and a maximum horizontal range of 30m while the 1800kHz is configured for half those values (0.1m vertical resolution and 15m horizontal range). Only the 900kHz channel was used when collecting data for the SMDD.

Figure 3.1: The Sea Scan® ARC Scout Mk II sonar sensor modules (black) and on-board processor unit (white).

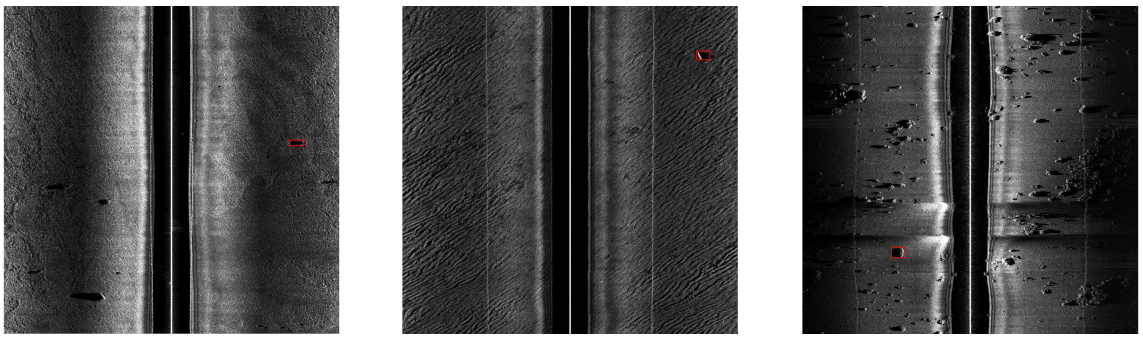
A REMUS 100 AUV was used as the mounting platform for the Scout Mk II sonar sensor. The REMUS 100 is a compact autonomous underwater vehicle designed for coastal ocean environments up to a depth of 100 meters. It can sustain a cruising speed of 1.5m/s for up 12 hours of continuous operation. The REMUS 100 has adaptive path planning and is capable of coordinating search operations in groups of four.

Figure 3.2: The REMUS 100 AUV in a deployment scenario. In this image the vehicle is traveling to the right and out of the page. The black bar on the side of the vehicle is the sonar sensor.

The dataset consists of three data categories: labelled images, unlabelled images and labelled snapshots. A brief outline of each category of data can be found in Table 3.1. Examples are shown in Figure 3.3 and 3.4.

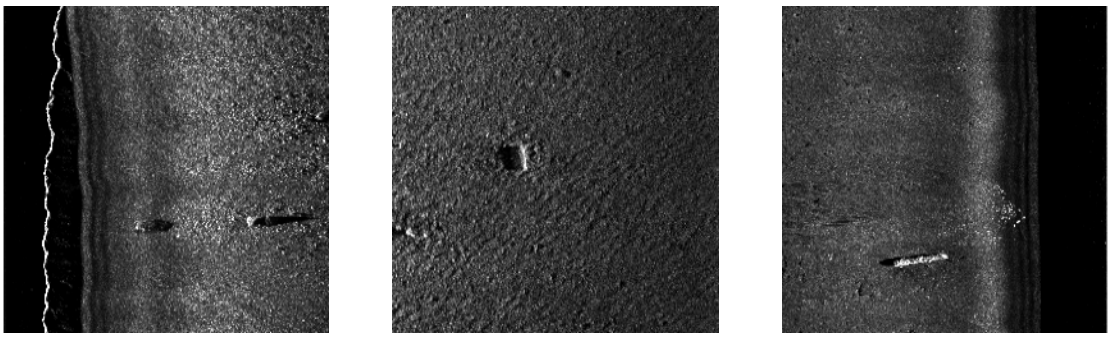
Table 3.1: An overview of the SMDD.

Data category	Total images	Total objects	Image resolution	Description
Labelled sonar images	152	198	1024×1000	Full size sonar images with at least one labelled object. There are 133 MLOs and 65 OSOs.
Unlabelled sonar images	11705	-	1024×1000	Full size unlabelled sonar images.
Labelled snapshots	196	116	101×101	Small sonar image crops centered around a labelled object (83 MLO and 33 OSO).



(a) (b) (c)

Figure 3.3: Three labelled sonar images from the SMDD. Labelled objects are outlined in red. The difficulty of seafloor environment is presented in ascending order. (a) Smooth sand seafloor, (b) Ripple sand seafloor, and (c) Sand seafloor with dense rock formations.



(a) (b) (c)

Figure 3.4: Three sonar snapshots from the SMDD. Each snapshot contains a labelled object at its center. These examples illustrate the variance of labelled MLOs. From left to right: (a) a semi-spherical MLO, (b) a cylindrical MLO parallel to sonar sensor and (c) a cylindrical MLO perpendicular to sonar sensor.

3.2 Proposed approach

The main focus of this research is underwater mine detection in which the main challenge is applying deep learning to real-world sonar image data. The obstacles we face with the SMDD are relevant to many other computer vision problems. It is, therefore, important that we outline the characteristics of sonar data that make it difficult to use deep learning, then move on to how we plan to address them.

- Small data: The number of labelled samples in our sonar dataset is very low relative to other deep learning tasks. This can lead to model overfitting.
- Under-representative data: The mine types and seafloor environments captured in the dataset are not representative of all possible scenarios. This means that the object representations formed by the network may not capture all MLO salient features.
- Covaritae shift: Imaging conditions greatly affect the quality of sonar images. This means that samples of identical scenes taken in the same location on separate occasions may not fall into the same data distribution.

Small data: Our dataset is comprised of 152 annotated sonar images which were provided by the DSTG. Compared to typical object detection datasets (see Table 3.1), this number is very low. The simplest method for generating additional image data is data augmentation. Initially, we investigate the effects of vertical and horizontal flipping when training sonar mine detection models. The SMDD is extended to 608 labelled images using only these two transformations. Figure 3.5 gives some examples of the vertical and horizontal flip transformations. Intensity transformations and more complex geometric transforms will be included in future experiments.

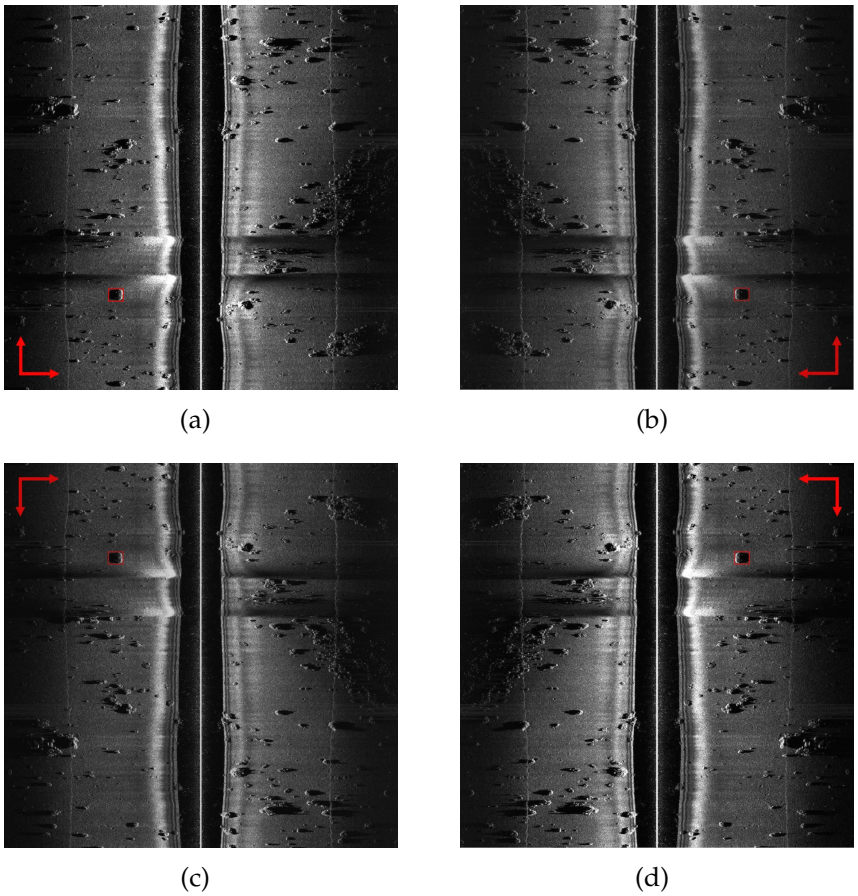


Figure 3.5: A sample sonar image (a) that has been: (b) horizontally flipped, (c) vertically flipped and (d) flipped both horizontally and vertically. The labelled object is outlined in red. The red arrows provide a frame of reference for the transformations

Next, we investigate transfer learning methods to adapt pretrained generic detection models to underwater mine detection. We use asymmetric transfer learning to train each model with weights that have been pretrained on the ImageNet detection task. Asymmetric transfer learning is chosen due to the difference in the source task and target task. Namely, generic detection models are trained with natural images while mine detection models deal with sonar images. A combined latent feature space will not be beneficial since detecting objects in the two tasks involve identifying very different high-level features.

Under-representative data: There are two challenges in terms of data representativeness with the sonar images. First, the diversity of seafloor environments is immense. The ocean makes up a majority of the earths surface, consisting of sand, rocks, clay and coral in different proportions. The SMDD does not have enough samples to adequately represent the diversity of the oceans. This is comparable to training a model that detects dogs on a grassy field during the day and expecting the model to perform well with dogs in the rain, at night, in the desert or underwater.

Second, the MLOs found in the SMDD dataset are not representative of a large portion of the MLOs found in the ocean. The ocean contains hundreds of varieties of underwater mines. These mines have distinct shapes, sizes and material compositions that lead to different highlights and shadows. Continuing on with the analogy of dogs, this is like training a model to detect only small brown dogs and attempting to detect all dog breeds.

In this research, we attempt to overcome the first problem of under-representative data pertaining to seafloor diversity. Models trained with a diverse dataset are less likely to make false detections in unknown environments. The SMDD has over 11,000 unlabelled sonar images. We propose an evolutionary data augmentation method to generate additional training images from unlabelled data and sonar snapshots. A detailed description of the algorithm is given in Section 3.3.

3.3 Evolutionary Data Synthesis

In this section, we provide a detailed explanation of our proposed data synthesis algorithm, called evolutionary cut-paste (ECP). The ECP algorithm generates additional sonar data by cutting objects from labelled sonar images and sonar snapshots then pasting the objects onto unlabelled seafloor environment. This approach has been used extensively in generic object detection [59–61] and more recently in underwater mine detection [12]. Our major contribution is framing cut-paste data synthesis as an optimisation problem and applying evolutionary algorithms as a search method.

3.3.1 Problem formulation

Given a set O of n labelled objects o_n and and a set U of m unlabelled sonar images i_m, generate a set S of n synthetic images s_n that minimised the validation loss L of a detection model M with loss function F:

$$L = \sum_{i=1}^{n} F(s_i) \tag{3.1}$$

The image generation constraints are:

- Each synthetic image s_i can only contain a single labelled object o_i.
- Each labelled object o_i must appear only once in the set of synthetic images S.
- Unlabelled images i_n are divided into a $W \times H$ uniform grid $G_{w,h}$ (see Figure 3.6).
- With $W = 7$, $H = 7$, the center column of the grid contains the nadir region and so it can be ignored. This functionally makes the grid $G_{w-1,h}$
- A labelled object o_i can only by overlaid at the center of a grid cell.

Given these constrains, the space of all synthetic images SS is:

$$SS = (W-1) \times H \times n \times m, \tag{3.2}$$

In our case, there are $n = 312$ labelled objects, $m = 11202$ unlabelled images and the grid has dimensions $W = 6$, $H = 7$. Therefore, the space of synthetic images SS is made up of $146,791,008$ possible images.

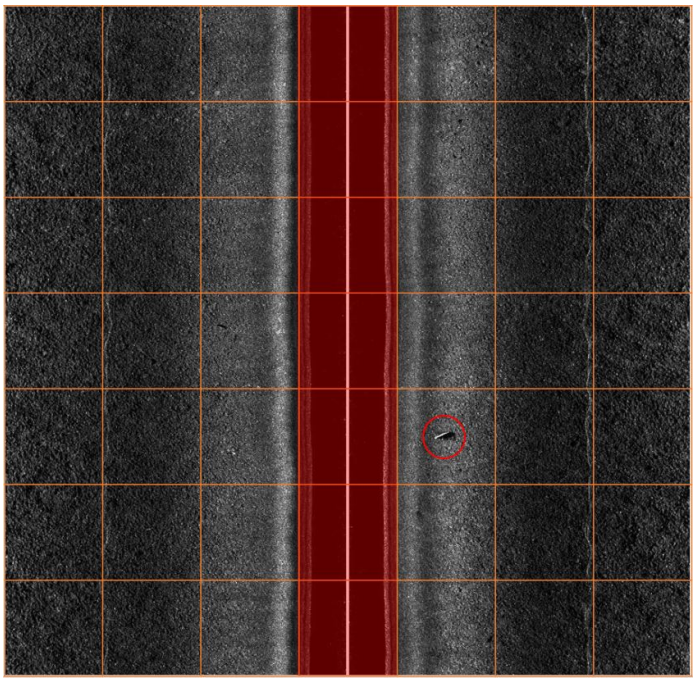


Figure 3.6: A synthetic sonar image with a 7×7 grid overlay. A MLO has been pasted in the center of the grid cell at column 5, row 5. The nadir region (row 4) is highlighted in red and is excluded.

3.3.2 Algorithm details

We propose to search the space of synthetic images SS with an evolutionary algorithm. In this section, we outline the main algorithmic steps. The pseudo-code of the ECP algorithm is given in Algorithm 1. Table 3.2 provides a brief description of evolutionary algorithm terms in our context.

Algorithm 1 ECP - evolutionary data synthesis algorithm

Start training epoch:
Set counter $c = 1$
Set totalGenerations = x
Generate initial population P of p individuals
while ($c < x$) **do**
    Train model M with training samples s_i of a single individual
    Evaluate individual based on validation loss L
    Select n fittest individuals Such that $n < p$
    Breed individuals with crossover
    Randomly mutate offspring
    Create new population Q w/ offspring and generate random individuals
    Increment c
end while

The ECP algorithm is a meta-heuristic optimisation algorithm in that it does not make any assumptions about the fitness landscape for our problem. Assumptions on which specific synthetic images contribute most to model performance are likely to be incorrect due to the stochastic nature of network training. For this reason, the ECP algorithm is more likely to converge to a good solution.

The search is initialised randomly upon the creation of the first generation of individuals. Subsequent generations are produced from selected individuals from the previous generation using three evolutionary mechanisms: crossover, mutation and recombination. Crossover involves breeding selected pairs of individuals by mixing their chromosomes randomly. The simplest method of crossover is single-point crossover. Alternatively, we can perform k-point crossover. Figure 3.7 shows examples of the crossover method. The ECP algorithm uses k-point crossover, where k is an integer in the range [1,10] and is chosen randomly before each crossover operation.

Mutations occur to individuals directly after crossover is performed. Mutation is a process that alters the chromosomes of individuals in a random way. Chromosomes are mutated randomly based on a variable that determines the likelihood of mutation. We define two operations for mutating a synthetic image: replacement and shifting. Replacement involves simply replacing the labelled

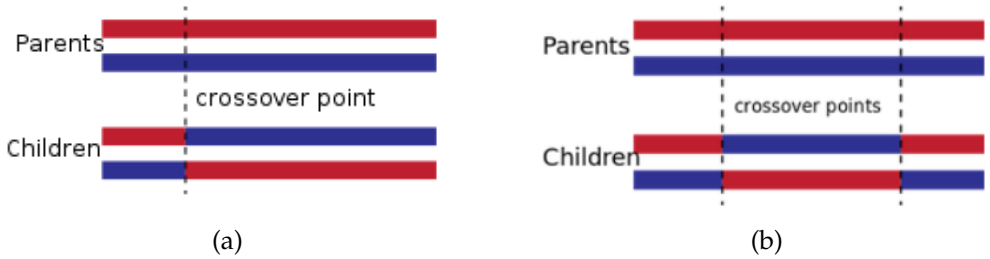


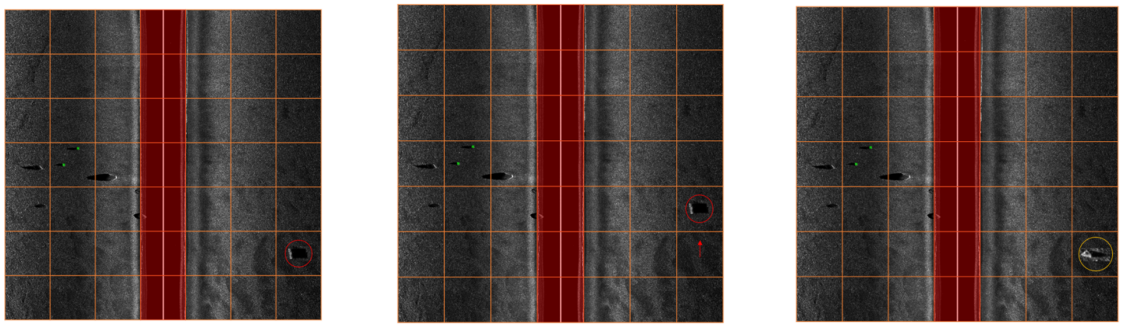
Figure 3.7: (a) Single-point and (b) two-point crossover between pairs of selected individuals. In this example, each pair produces two children.

object in the image with another object from the SMDD. Shifting involves moving the labelled object in the synthetic image randomly to an adjacent grid cell. The goal of mutation is to extend the search to neighbouring regions of the search space.

Table 3.2: Evolutionary algorithm terms defined in the context of data augmentation.

Term	Definition
Chromosome	A single synthetic image.
Individual	A set of synthetic images that are evaluated together.
Generation	A group of individuals that can breed and mutate to create the next generation.
Population	The entire set of individuals across all generations.

In biological evolution, recombination is the exchange of genetic material between two individuals that produces a trait in the offspring that is not present in either of the parents. We propose incorporating recombination into the ECP algorithm by splicing images together. Since all side-scan sonar images are divided by a nadir region, we can freely mix and match the left half of an image with the right half of a different image.



(a) (b) (c)

Figure 3.8: An example of mutation applied to a synthetic image. (a) Original synthetic image, (b) Synthetic image mutated with shifting, (c) Synthetic image mutated with replacement.

3.4 Experiment details

3.4.1 Introduction

We conducted preliminary experiments in underwater mine detection. There are four main goals:

- To compare the performance of current state-of-the-art detection algorithms.
- To determine the applicability of pretrained generic detection models.
- To measure the impact of data augmentation on model detection accuracy and overfitting.
- To evaluate our proposed data synthesis algorithm.

3.4.2 Dataset

We performed experiments with the SMDD. The SMDD consists of labelled and unlabelled sonar images as well as smaller labelled snapshots. The dataset has two labelled object classes: MLOs and other significant objects (OSOs). Table 3.1 gives an overview of the SMDD. For our experiments, we use 4-fold cross-validation. The labelled images are divided into four folds which corresponds to

splitting the dataset into 75% train and 25% test data. No validation set is used since the experiments do not involve any hyper-parameter tuning.

3.4.3 Evaluation metrics

We used MS COCO detection metrics [33] for evaluating model performance. The accuracy of individual predictions are based on intersection over union (IoU). IoU is a measure of overlap between two bounding boxes, as shown in Figure 14. Detections are grouped into one of three categories: A true positive (TP), false positive (FP) and false negative (FN).

- A TP occurs when a prediction specifies the correct class for an object and has an IoU score higher than the predefined threshold.
- A FP occurs when a prediction specifies the incorrect class for an object.
- A FN occurs when an object is not detected when it should have been.

Figure 3.10 contains an example of each category. Only the bounding box with the highest IoU score is counted as a TP if multiple boxes detect the same object. The remaining boxes are classified as FPs.

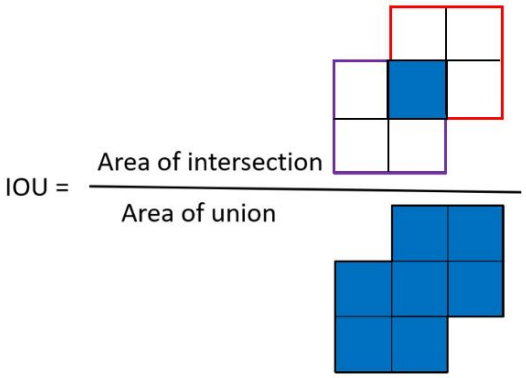


Figure 3.9: A visual example of the IoU calculation. In this example, the IoU score will be $\frac{1}{7}$ since the overlapping region is one-seventh the size of the total area made by the two boxes.

The two main indicators of detection performance across an entire dataset are mean average precision (mAP) and mean average recall (AR). The metrics are more advanced variants of precision (P) and recall (R) that treat each object class independently. Precision and recall are defined mathematically as

$$P = \frac{TP}{TP + FP} \ , \tag{3.3}$$

$$R = \frac{TP}{TP + FN} \ , \tag{3.4}$$

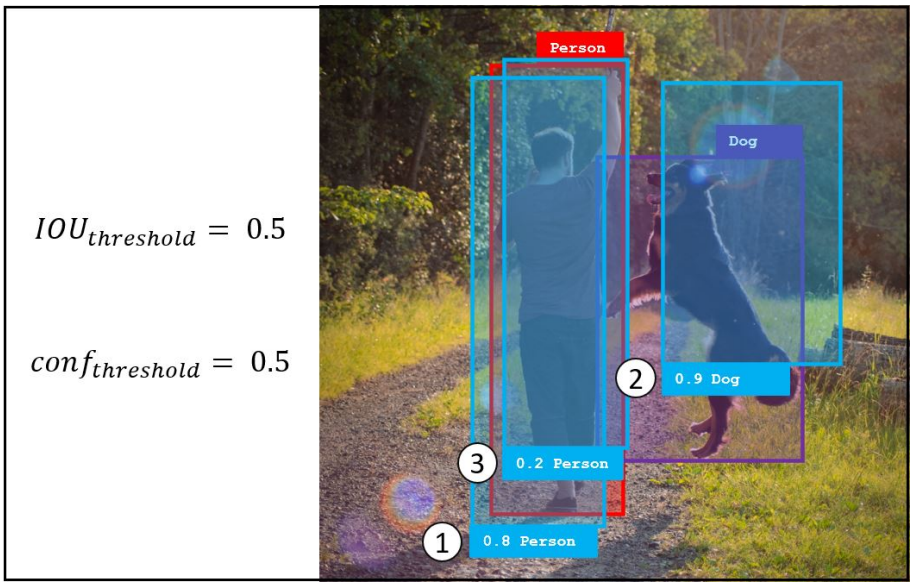


Figure 3.10: An image containing two objects (one person and one dog) with ground-truth labels and bounding boxes drawn (red and purple). In this example the detector model has made three predictions, shown in blue and numbered. Prediction 1 is a TP. Prediction 2 is a FP. Prediction 3 is a FN.

3.4.4 Experimental method

We compared four state-of-the-art detection algorithms for underwater mine detection. For YOLO, an implementation from GitHub was used [62]. For all other algorithms, Tensorflow object detection API was used [63].

- **YOLOv3** [41]: Single-stage detector with DarkNet-53 as the backbone.
- **Tiny YOLOv3** [41]: Single-stage detector with DarkNet-19 as the backbone.

- **Faster R-CNN** [48]: Two-stage detector with ResNet-50 as the backbone.
- **SSD** [42]: Single-stage detector with ResNet-50 as the backbone
- **EfficientDet-d1** [43]: Single-stage detector with ResNet-50 as the backbone.

Each model was trained with 4-fold cross-validation in one of four ways: (1) from scratch without data augmentation, (2) from scratch with data augmentation, (3) transfer learning without data augmentation, (4) transfer learning with data augmentation. We chose only two simple image transformations, horizontal flip and vertical flip, to simplify the experiment and enable easy performance comparison between different training approaches.

3.4.5 Network training

The models are trained for 100 epochs per fold. Default learning rates as per the TensorFlow object detection API [63] were used when transfer learning. For training from scratch, the learning rates were reduced by a factor of four to avoid unstable gradients. A detailed description of the complete training configuration for each model can be found in Appendix A.

3.5 Results

3.5.1 Transfer learning

Results: Table 3.3 and 3.4 present the performance of the models based on mAP and mAR with an IoU threshold of 0.5. Scores are averaged across the four folds. The performance of every detection model was improved by both data augmentation and transfer learning in each training configuration, as shown in Figure 3.12. We can process the values in Table 3.3 and 3.4 to conclude that data augmentation improved the tested models mAP by 16.9% and mAR by 21.1% while transfer learning lead to an increase of 11.9% in mAP and 21.1% in mAR.

This result validates the use of these techniques in future experiments with sonar data.

Overall, Tiny YOLOv3 achieved the best mAP and mAR scores out of the five models except for in the transfer learning mAR metric, where Faster R-CNN achieved 0.3204 mAR with data augmentation. This indicates that the pre-trained weights for the region proposal network of Faster R-CNN may be useful in identifying key regions in sonar images. We use Tiny YOLOv3 in the next experiment on evolutionary data augmentation.

Table 3.3: The mAP @ 0.5 IoU scores of the detection models

	Train from scratch		Transfer learning	
Model	No data augment	Data augment	No data augment	Data augment
YOLOv3 [41]	0.0421	0.1124	0.06153	0.1756
Tiny YOLOv3	**0.0671**	**0.1461**	**0.0972**	**0.2143**
Efficientdet d1[43]	0.0246	0.0943	0.0671	0.1912
Faster R-CNN [48]	0.0142	0.0505	0.0966	0.1936
SSD [42]	0.0336	0.1163	0.0728	0.1300

Table 3.4: The mAR scores of the detection models

	Train from scratch		Transfer learning	
Model	No data augment	Data augment	No data augment	Data augment
YOLOv3 [41]	0.0724	0.1623	0.1277	0.2514
Tiny YOLOv3	**0.0813**	**0.221**	0.1642	0.2882
Efficientdet d1 [43]	0.0621	0.1549	0.1136	0.2092
Faster R-CNN [48]	0.0712	0.1051	**0.2121**	**0.3204**
SSD [42]	0.0766	0.1936	0.1365	0.2674

Our results indicate that a smaller network is suitable for the task of mine detection. We suggest that the reduced number of parameters prevents overfitting and encourages the learning of more generalisable features. EfficientDet has the lowest precision and recall out of the single stage detectors when trained from scratch, suggesting that the pre-trained weights for the bi-directional feature pyramid are beneficial. We also note that the region proposal network in Faster-RCNN requires additional training and should only be used with pre-trained weights from a large detection dataset. Given these results, we are motivated to incorporate a bi-directional feature pyramid in the Tiny YOLOv3 architecture. Figures 3.17 and 3.18 give some visual examples of the Tiny YOLOv3 model outputs.

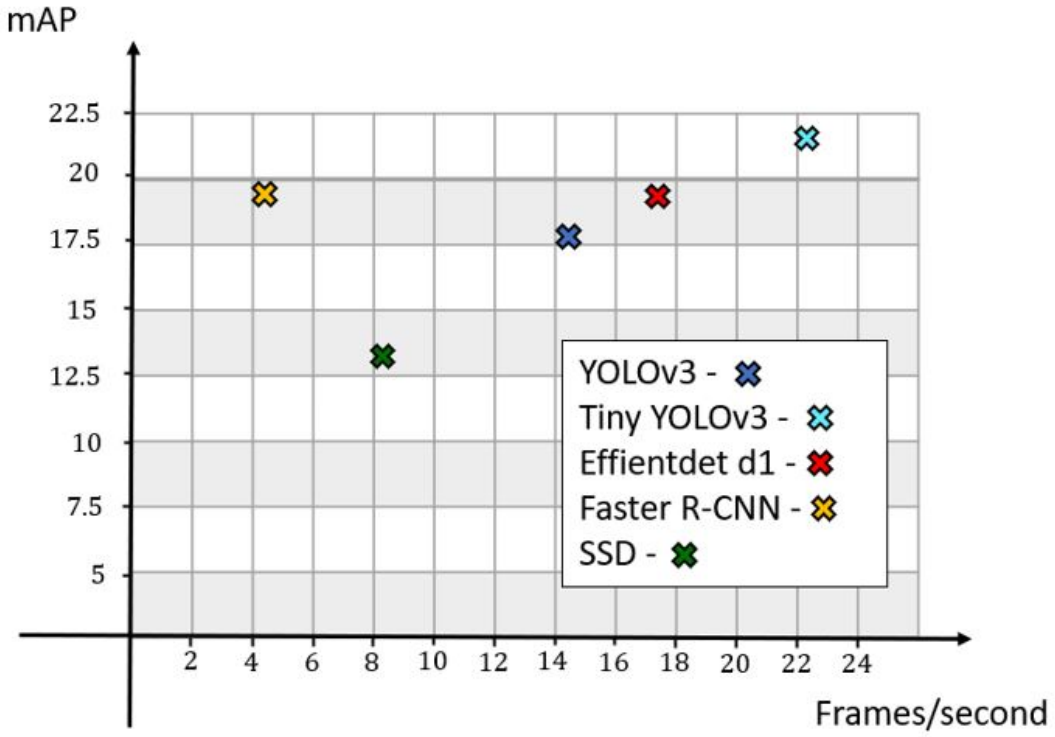


Figure 3.11: A comparison of mAP and frames per second of the models. Tiny YOLOv3 has the fastest inference time (46ms) and highest precision (21.43%).

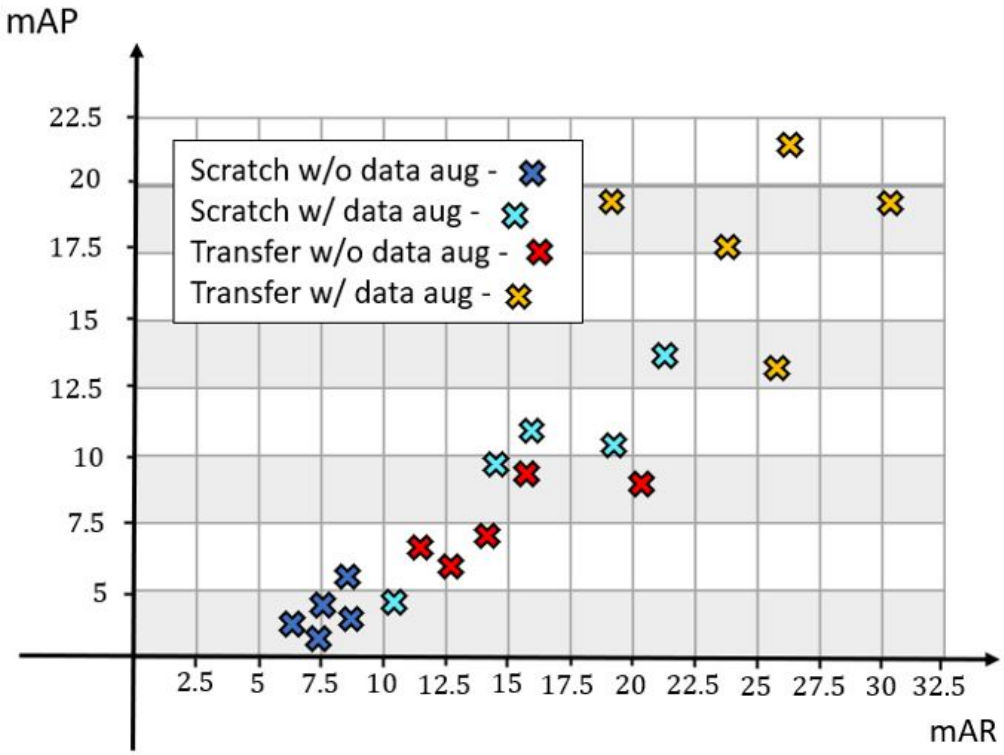


Figure 3.12: A plot showing the mAR and mAP of the detection models with different training configurations. As expected, models trained from scratch with no data augmentation performed poorly. Both transfer learning and data augmentation lead to an improvement, however transfer learning had a greater benefit. Models trained with transfer learning and data augmentation performed on average 4 times better than the baseline.

3.5.2 Data synthesis experiment

In this experiment, we evaluate our proposed evolutionary data synthesis method. The aim is to compare the performance of evolutionary selected sets of synthetic images and randomly selected synthetic images against a baseline of no synthetic images. To do this, we first generate four sets of synthetic images with the proposed augmentation algorithm from Section 3.3.1. Next we train a Tiny YOLOv3 model with the same configuration as in the previous experiment (transfer learning with data augmentation). The model will be trained eight times: four times with evolutionary synthetic images (one time for each set), and four times with randomly generated synthetic images.

Data synthesis: We use the ECP algorithm from Algorithm 1 in Section 3.3.2 to

generate four sets of synthetic images. The algorithm is run for 40 generations with each generation containing 16 individuals. The eight fittest individuals from each generation are selected and randomly sorted into pairs. K-point crossover is performed three times on each couple which yields a total of 12 individuals for the next generation. For our experiment, we choose k to be an integer in the range [1,10]. The remaining four individuals needed for the next generation are generated randomly. In this experiment, we exclude 25% of the SMDD from the ECP algorithm to avoid including objects from the test set into the data synthesis process.

Fitness evaluation: Individuals are evaluated after three epochs of training a YOLOv3 Tiny model with synthetic images. The training is initialised with a checkpoint obtained in the transfer learning experiment from Section 3.5.1. The checkpoint is at the point where training and validation loss diverge. We use this checkpoint instead of training from scratch to avoid variance caused by randomly initialised model weights. At the end of the three training epochs, the models are shown a test set to calculate the loss.

Our initial results indicate that the ECP algorithm is effective at finding beneficial sets of synthetic images. The average fitness scores of individuals decreases with each generation. This indicated that the sets of synthetic individuals are improving as the algorithm progresses.

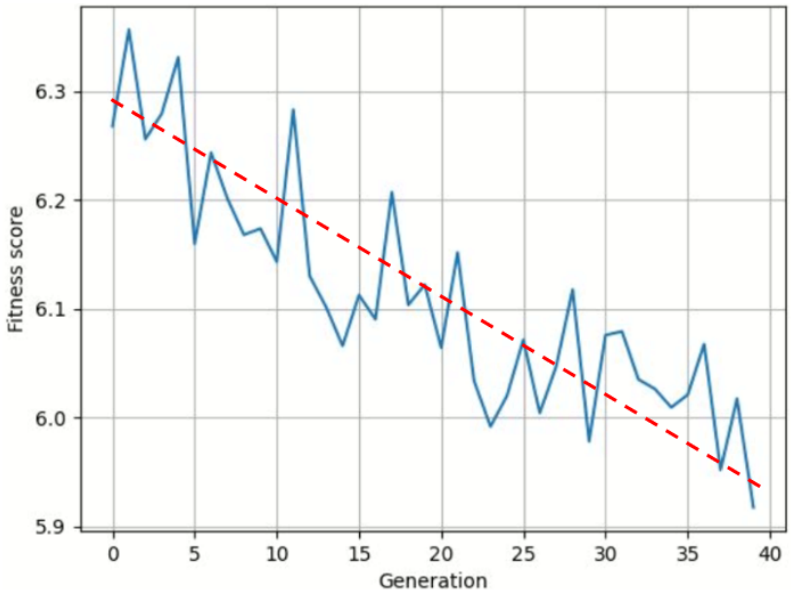


Figure 3.13: Mean individual fitness across subsequent generations. The fitness score is based on validation loss (lower is better). We observe a clear downward trend over 40 generations.

It can be observed from Figure 3.14 that useful synthetic images (chromosomes) propagate through the population by tracking chromosomes across generations. We refer to the number of generations that a chromosome has propagated through as the age of the chromosome. For example, if a certain chromosome has been selected during reproduction from generation 12 to generation 15, we say that the chromosome has an age of 3. The difference between the mean and the medium age curve is caused by resilient sets of chromosomes that survive for many generations. It is interesting to note that a small group of images propagated over 38 generations.

Next, we discuss the effect of synthetic images on model detection performance. Figure 3.16 shows the precision-recall curve of the Tiny YOLOv3 models trained with synthetic images. The curves are averaged across four runs of evolved and random synthetic images. We also include a baseline of no synthetic images for comparison. The models are evaluated based on the area under the curve (AUC) of the precision-recall curve. The models trained with evolutionary synthetic images perform consistently better across training runs with an AUC

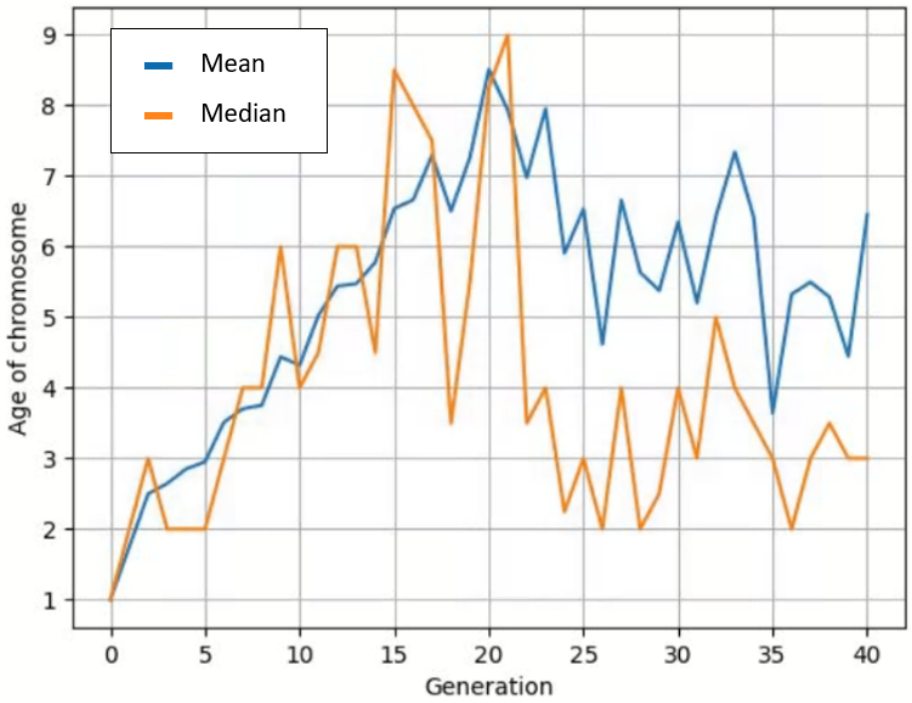


Figure 3.14: Mean and median chromosome age across subsequent generations. The mean age of chromosomes in the population settled higher than the median age after initially increasing at the same rate. This is due to resilient chromosomes that last through many generations while new chromosomes are introduced to the gene pool.

value of 0.2471. This value is 6.86% higher than the AUC of the model trained with random synthetic images and 20.2% higher than the model with no synthetic images. The result supports our hypothesis that evolutionary algorithms are a viable method for synthesising additional data.

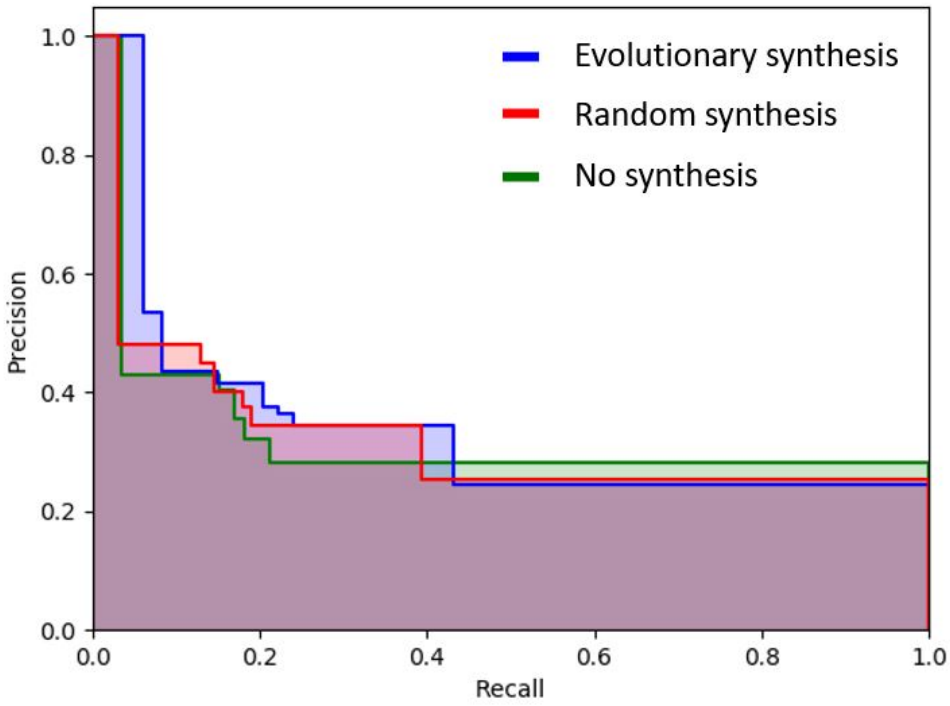


Figure 3.15: Precision-Recall curve for the three models.

Finally, sample visual results of underwater mine detection are shown in Figures 3.17 and 3.18. The trained detector (Tiny YOLOv3) works reasonably well on the sand seabed. It still produces some false detections, which could be eliminated by training the detector with more labelled sonar images.

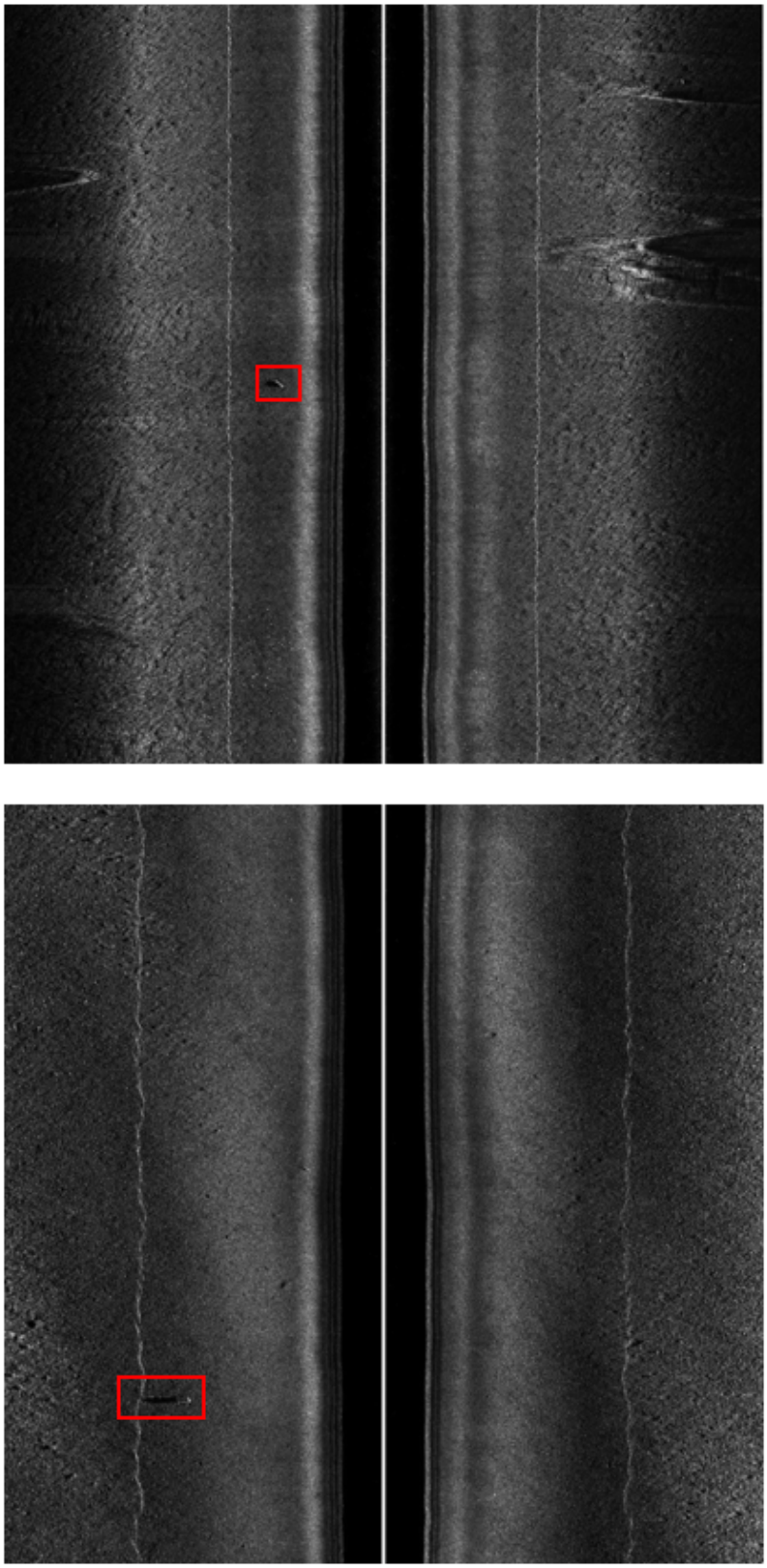
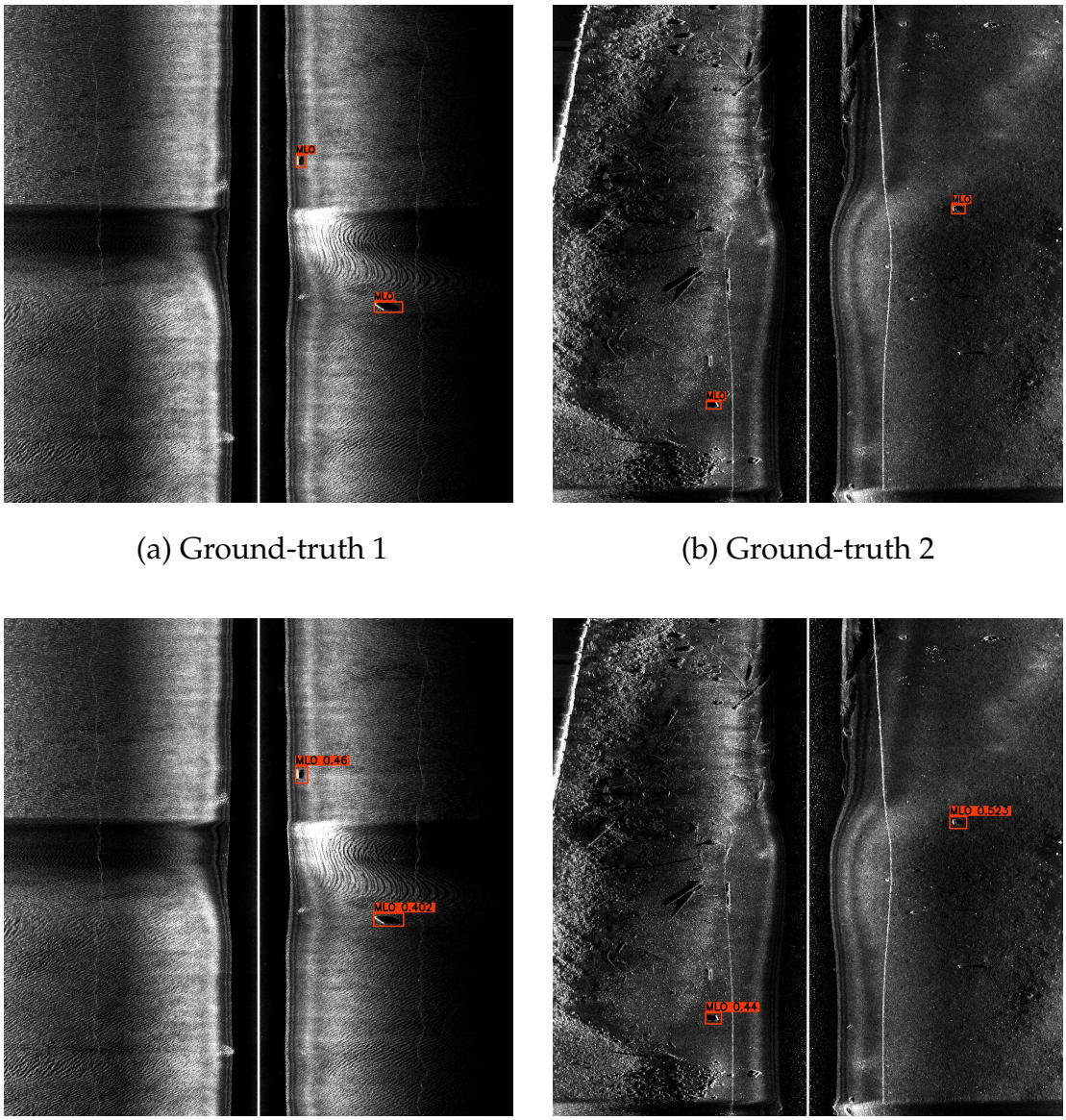


Figure 3.16: Two synthetic images (chromosomes) that survived for 37 generations. Bounding boxes are left loose so as to not occlude the object boundary. The objects blend well into the scene which makes for a realistic synthetic image.



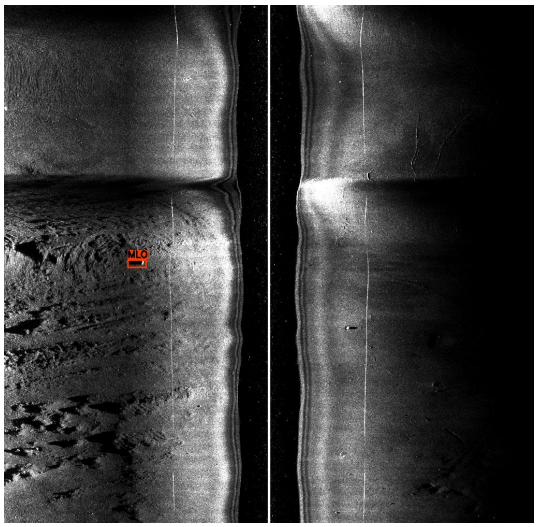
(a) Ground-truth 1

(b) Ground-truth 2

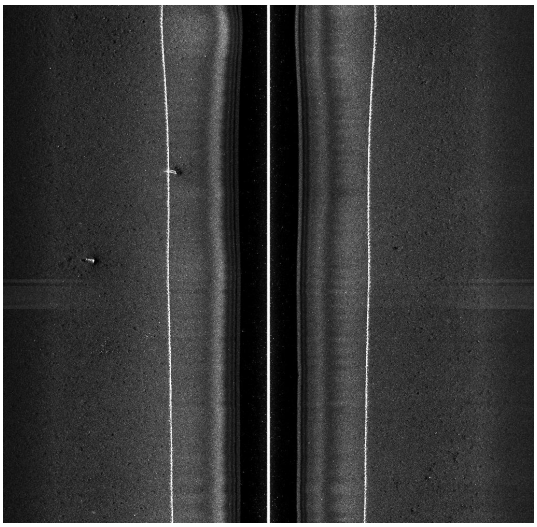
(c) Tiny YOLOv3 output of (a)

(d) Tiny YOLOv3 output of (b)

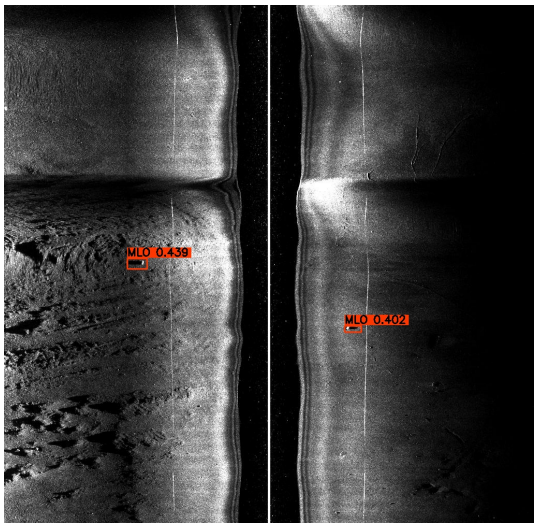
Figure 3.17: Visual results of underwater mine detection produced by Tiny YOLOv3. *Top row*: Sonar images with manually annotated MLO ground-truths. *Bottom row*: Detection predictions of Tiny YOLOv3.



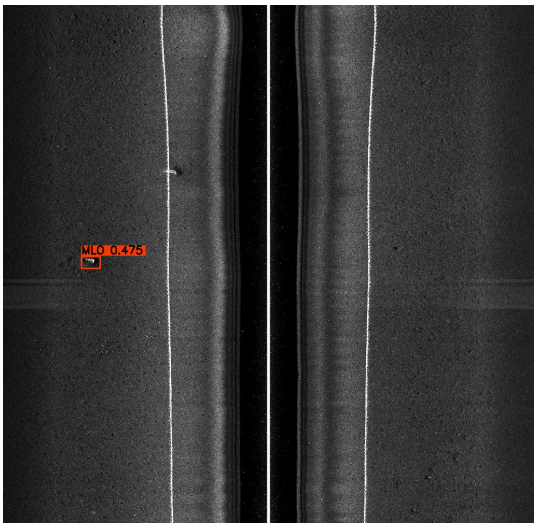
(a) Ground-truth 3



(b) Ground-truth 4



(c) Tiny YOLOv3 output of (a)



(d) Tiny YOLOv3 output of (b)

Figure 3.18: Further visual results of underwater mine detection produced by Tiny YOLOv3. *Top row*: Sonar images with manually annotated MLO ground-truths. *Bottom row*: Detection predictions of Tiny YOLOv3.

Chapter 4

Seabed segmentation

Chapter contents

4.1 Segmentation dataset overview 48
4.2 Data annotation 50
4.3 Proposed approach 51
4.4 Experimental details 56
4.4.1 Introduction 56
4.4.2 Dataset 57
4.4.3 Evaluation metrics 57
4.4.4 Experimental method 58
4.5 Results 59
4.5.1 Learning rate range test 59
4.5.2 Cyclic and OneCycle learninig rate schedules 65
4.5.3 Median frequency balancing 68
4.5.4 Comparing encoders 70
4.5.5 Comparing architectures 73

Our second objective is to investigate SOTA segmentation models and encoders in the seabed segmentation task. In section 4.1 we give a detailed overview of our segmentation dataset. In section 4.2 we outline the process used for data annotation. In section 4.4 we cover specific details regarding our experiments. In section 4.5 we discuss results.

4.1 Segmentation dataset overview

Understanding the composition of the seabed is critical in underwater surveying tasks. We identified nadir, sand, sand ripple, and rock to be the four predominant types of regions in the SMDD. As such, We created a new dataset, called the seabed segmentation dataset (SBSD). The SBSD is an extension of the SMDD (discussed in Section 3.1) that includes images from the SMDD plus other previously unlabelled samples. We purposefully excluded some images from the SMDD that were mostly sand to avoid over-representation in the dataset. From the full-size annotations, we created an additional dataset of non-overlapping image patches. An overview of the SBSD is give in Table 4.1. Reasoning behind the decision to make an additional image patch dataset will be discussed in Section 4.4.

Table 4.1: An overview of the SBSD.

Data category	Total images	No. Classes	Image resolution	Description
Full-size images	137	5	1024×1000	Full size sonar images.
Patch images	2,192	5	256×256	Non-overlapping patches

For each image, we produced a corresponding pixel-wise ground-truth mask. Each pixel in the image is assigned a value which corresponds to the class that the pixel belongs to. The masks can be visualised by colour coding the pixels of each class. Some examples from the dataset can be seen in Figure 4.1.

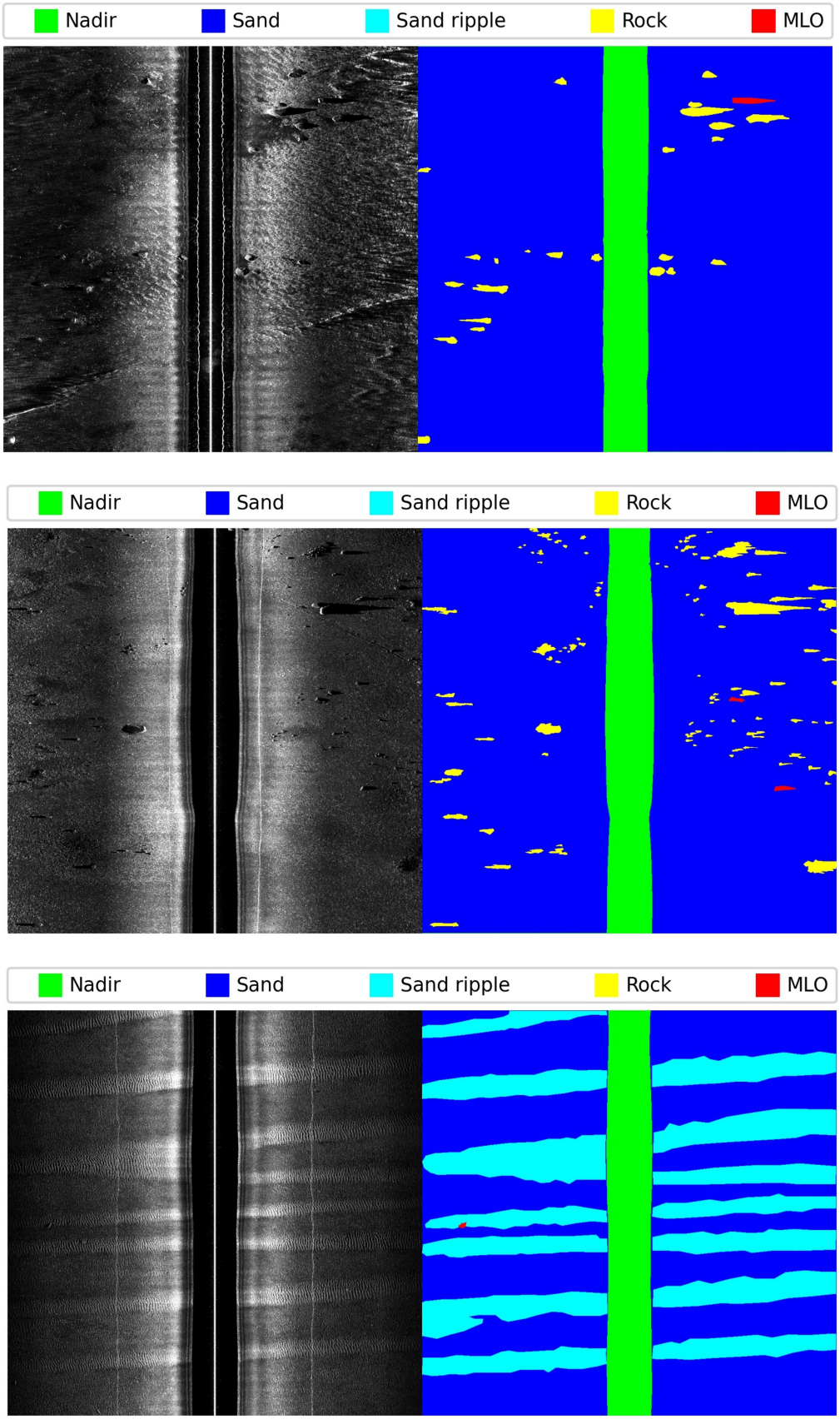


Figure 4.1: Three annotated images from the SBSD.

4.2 Data annotation

We used an open-source software tool called LabelMe for data annotation. The tool provides a GUI that lets us draw polygons over a sample image. Each polygon is assigned a class that classifies the pixels contained within the polygon. Other useful features include image zoom, contrast and saturation sliders. Figure 4.2 shows the LabelMe GUI with an example image.

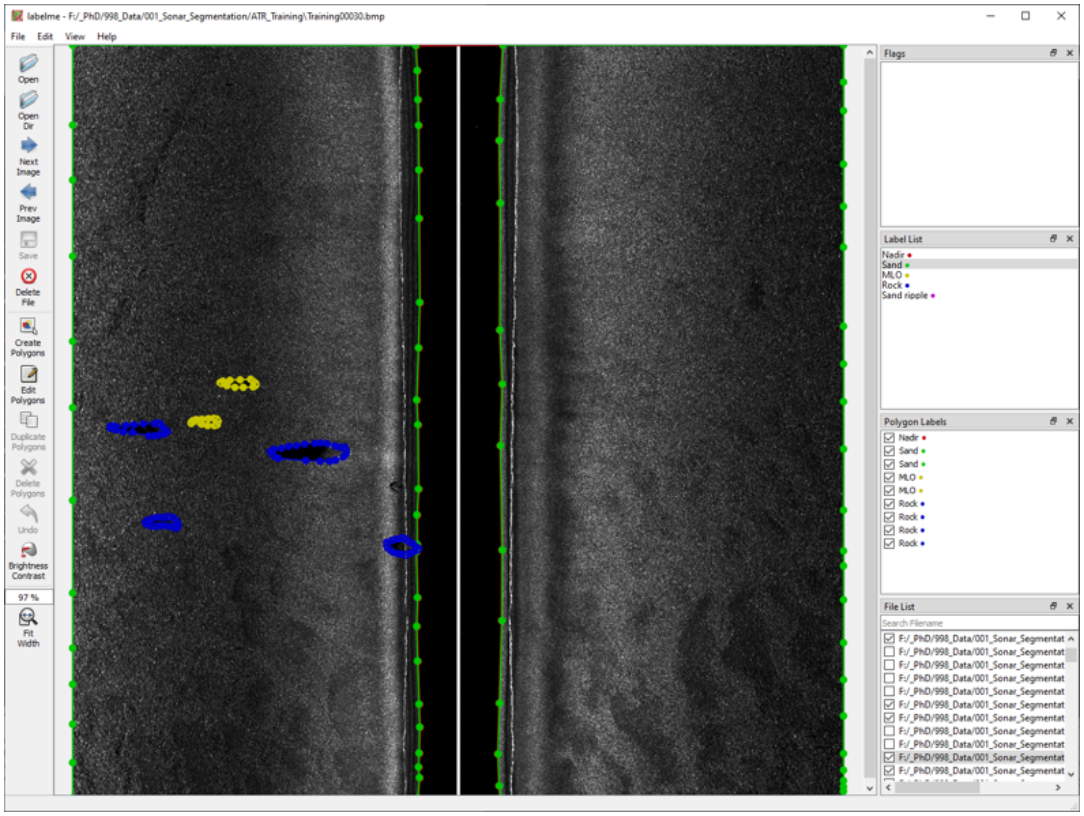


Figure 4.2: LabelMe UI displaying an annotated sonar image.

LabelMe exports the ground-truth masks in JSON format. The tool provides an example function that converts a single JSON to a PNG image, but does not natively support large-scale conversions. Hence, we developed multiple auxiliary files to automate data conversion.

4.3 Proposed approach

The goal of our experiment is to compare SOTA segmentation architectures and encoders in sonar image segmentation. Furthermore, we use various techniques to optimise the performance of our models. The techniques will be discussed in the paragraphs to follow. Given the number of model configurations we are testing, it was crucial to ensure that good hyper-parameters were found prior to evaluating the models. Leslie and Topin [64] proposed a novel hyper-parameter optimisation technique called the learning rate range test (LRRT). The LRRT is used to find optimal values for learning rate and weight decay for model configurations.

The LRRT involves training a model with linearly increasing hyper-parameters over a small number of training steps, typically less than 10% of a full training run. A wide range of values for each hyper-parameter are selected prior to initialising the test. The range is chosen to start from a small value, one that hardly allows the model to begin converging, up to a large value that you would not normally consider to use for training. For example, if we know that 5×10^{-4} is a "good" learning rate for a generic U-net model, we would test values from 5×10^{-5} to 0.5. The goal of the test is to find the point where gradients explode. Figure 4.4 gives an example of exploding gradients. This indicates that the hyper-parameter has grown to be too large. A more suitable learning rate can be chosen based on this information.

The hyper-parameters found by the LRRT are applied by using either the OneCycleLR [64] scheduler or cyclic learning rates [65]. The OneCycleLR scheduler leverages high learning rates as a form of regularisation. At the beginning of training, the model is trained with a small learning rate, min_lr. The learning rate increases linearly after each training step until it reaches its maximum value, max_lr. At this point, the learning rate decreases linearly back down to the min_lr. Finally, the learning rate is reduced further in what is called the annihilation phase, where the learning rate approaches zero. The annihilation phase

allows the model to converge to a local minima. An example of the OneCycleLR is shown in Figure 4.3. It is important to note that the duration and magnitude of each phase can be adjusted.

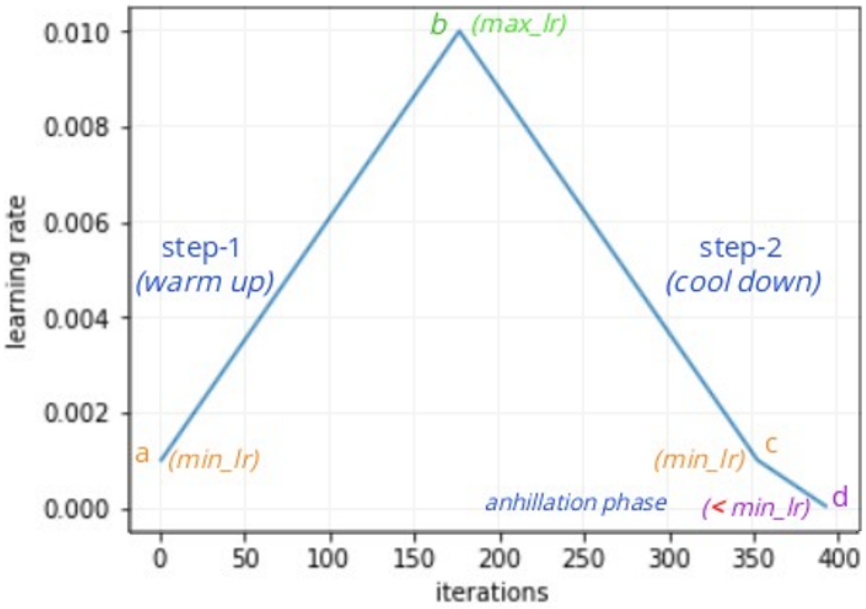


Figure 4.3: The OneCycleLR scheduler.

Six encoders were chosen for testing. The encoders are MobileNet_v2 [66], EfficientDet-d0 [43], ResNet18 [67], DPN68 [68], VGG13_bn [69] and DenseNet18 [70]. For each encoder, we found optimal values for the learning rate and weight decay. This involves performing two runs of the LRRT for each encoder. The first run attains a value for the learning rate, and the second gives us the corresponding weight decay. The experiment uses the same model architecture, U-Net, with each of the encoders. All other parameters are kept the same.

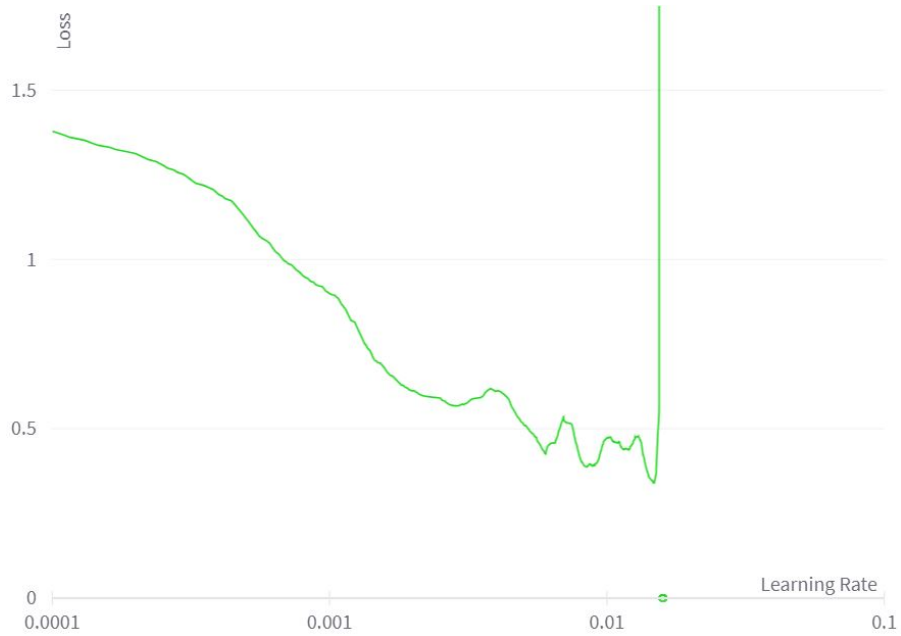


Figure 4.4: An example of exploding gradients.

Next, we investigate deep learning optimiser algorithms. Optimisers play an important role in DNN training. Several algorithms such as Adam [71], Adagrad [72], and Rmsprop have become increasingly popular in the past. These algorithms improve upon stochastic gradient descent (SGD) by adapting training parameters after each step of training. Smith and Topin [64] discovered that models can be trained faster and with better performance by combining the SGD and OneCycleLR schedule. We compare popular optimisers with the previously discussed method in section 4.5.2. We will briefly discuss each algorithm before moving to the main point of this section.

Stochastic gradient descent is the main approach for training deep neural networks. First introduced by Robbins and Monro [73], it updates the weights of the network depending on the loss after each training step. The weights θ of a DNN are updated to minimise an objective function $J(\theta)$, which is computed from input-output pairs (x^j, y^j). Learning rate α directly scales magnitude of the change as

$$\theta_{i+1} = \theta_i - \alpha \times \nabla_\theta J(\theta; x^j; y^j). \tag{4.1}$$

The Adagrad (adaptive gradient) algorithm [72] improves SGD by replacing the fixed learning rate with a per-parameter learning rate. Weights are updated based on how much change has occurred in previous iterations. This is achieved by taking an element-wise product of the objective function $J(\theta)$ and a vector G. G is defined as the sum of the outer product of gradients g for steps up to t:

$$\mathrm{G} = \sum_{\tau=1}^{t} g_\tau g_\tau^\intercal. \tag{4.2}$$

The weights of the network, θ, are updated after every iteration as

$$\theta_{i+1} = \theta_i - \alpha \ \ \mathrm{diag}(G)^{\frac{-1}{2}} \odot g. \tag{4.3}$$

RMSprop is an extension of Adagrad first proposed by Geoff Hinton. It builds on top of Adagrad by using the decaying exponential moving average of the partial derivatives of network weights rather than the sum. A new coefficient, ρ, is introduced which affects the momentum of the moving average. The moving average of squared gradients is calculated as follows:

$$\mathrm{E[g^2]}_t = \rho E[g^2]_{t-1} + (1-\rho)\frac{\delta L}{\delta W}^2, \tag{4.4}$$

The moving average of the squared gradients, $E[g^2]$, is then used to scale the magnitude of the change in gradients for the next step:

$$\mathrm{w}_t = w_{t-1} - \frac{\eta}{\sqrt{E[g^2]}}\frac{\delta L}{\delta W}. \tag{4.5}$$

The Adam (adaptive moment estimation) optimiser [71] makes use of a running average of first and second-order gradients to speed up training convergence. It combines the benefits of the two previously mentioned optimisers and controls the decay using two momentum constants, β_1 and β_2. The first- and second-order gradients, m_t and v_t, are calculated as in Equation 4.4.

Bias correction and momentum are factored into the moving averages as

follows:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1}. \tag{4.6}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2}. \tag{4.7}$$

A very small constant, ϵ, is used to avoid division by zero. Finally, the network weights are updated as

$$\mathrm{w}_t = w_{t-1} - \alpha \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} - \epsilon}. \tag{4.8}$$

Next, we focus on the problem of class imbalance in the SDD. MLOs and rocks occur much less frequently than the three classes. Additionally, they are visually very similar to each other. This results in two unwanted consequences. The first problem is that the model has less data to train with for segmenting the under-represented regions. Secondly, the dominance of rock occurrence over MLO occurrence leads to the model segmenting many MLO regions as rocks due to their similar appearance. An overview of the frequency of pixels for each class can be found in Table 4.2.

Table 4.2: Class distribution in the SSD.

Class	**Total pixels**	**Percentage of dataset (%)**
Nadir	16,301,601	11.62
Sand	93,530,325	66.67
Sand ripple	29,222,092	20.83
Rock	1,192,560	0.85
MLO	41,422	0.03
Total	140,288,000	100

Median frequency balancing (MFB) [50] helps to minimise the tendency of models to bias their outputs towards more common classes. We propose using median frequency balancing (MFB) as a way to counteract class imbalance. MFB is a technique for scaling the impact specific classes have on back propagation. This is achieved by adding a coefficient to scale each class's contribution to the loss function. The coefficients are determined by considering each classes frequency in the dataset compared to the median class, hence the name median frequency balancing. The collection of these coefficients is called a weight tensor. MFB produces a weight tensor W as

$$\mathrm{w}_i = \frac{N_m}{N_i}, \tag{4.9}$$

where w_i is the scaling coefficient of the i-th class, N_m is the number of pixels belonging to the median class, N_i is number of pixels belonging to the i-th class.

The final stage of our experiment is a comparison of SOTA segmentation architectures and encoders. We start by evaluating the U-net with six different encoders. Next, we choose the best performing encoder as the baseline when comparing segmentation architectures. We use optimal hyper parameters from previous experiments.

4.4 Experimental details

4.4.1 Introduction

We conducted experiments in seabed segmentation. There are four main goals:

- To compare the performance of current state-of-the-art segmentation algorithms and encoder architectures.
- To evaluate the LRRT for hyperparameter optimisation.
- To evaluate alternate learning rate schedules for model training.
- To investigate methods for overcoming data imbalance.

All our code was written in Python and used the Pytorch deep learning library. Additionally, we used pretrained models from the segmentation_models Pytorch repository [74].

4.4.2 Dataset

The experiments in this section were conducted using the SBSD patch dataset (See Section 4.1). We used the patch dataset for three reasons.

1. Dividing the images into patches increases the total number of samples available.

2. The segmentation models that we tested are not suitable for 1000×1024 px images.

3. A larger number of patches can be batched together which speeds up training and improves regularisation.

4.4.3 Evaluation metrics

We use standard segmentation evaluation metrics to evaluate the outcomes of our experiments. The metrics are pixel accuracy, mean accuracy, intersection over union (IoU or Jaccard index), and F1 score (Dice score). Additionally, we will compare the inference speed and overall model size.

Pixel accuracy is the ratio of correctly classified pixels over the total number of pixels in a segmentation mask. Let n_{ij} be the number of pixels of class i that are classified as class j. Furthermore, let t_i be the total number of pixels of class i. Pixel accuracy is calculated as

$$\text{Pixel accuracy} = \frac{\sum_i n_{ij}}{\sum_i t_i}. \tag{4.10}$$

Mean accuracy is similar to pixel accuracy but the accuracy of each class is treated separately. The pixel accuracy score for each class is summed and then

divided by the total number of classes N as follows:

$$\text{Mean accuracy}= \frac{1}{N}\sum_{i=0}^{N}\frac{\sum_i n_{ii}}{\sum_i t_i}. \tag{4.11}$$

Furthermore, per-class accuracy adds weights to the less frequent classes, and therefore is a valuable indicator in applications with imbalanced classes.

IoU (Jaccard index) is a measure of overlap between two sets. In segmentation, we have two sets A and B which correspond to the ground-truth mask and predicted mask, respectively. Intuitively, the IoU describes the proportion of overlapping between two regions (intersection) relative to their total area (union):

$$IOU = J(A,B) = \frac{|\,A \cap B\,|}{|\,A \cup B\,|}. \tag{4.12}$$

Equation 4.12 is suitable for evaluating a binary segmentation output. We can extend the equation to multi-class segmentation by calculating the mean IoU across all classes as follows:

$$mIOU = \frac{1}{N}\sum_{i=0}^{N}\frac{|\,A_i \cap B_i\,|}{|\,A_i \cup B_i\,|}. \tag{4.13}$$

F1 score (Dice score) is similar to IoU, except that the nominator term (intersection) is multiplied by 2:

$$F_1(A,B) = \frac{2\,|\,A \cap B\,|}{|\,A \cup B\,|}. \tag{4.14}$$

The two metrics F1 score and IOU are positively correlated.

4.4.4 Experimental method

We begin by giving a brief outline of the experimental steps, listed in chronological order of how they are performed. In section 4.5 we will go into more detail on each of the steps.

1. Find optimal learning rate and weight decay for our selected encoders and a baseline segmentation architecture (U-Net).

2. Investigate five learning optimiser configurations.

3. Evaluate median frequency balancing and data augmentation.

4. Compare deep learning segmentation architectures and encoders.

4.5 Results

4.5.1 Learning rate range test

Firstly, we investigated the usefulness of the LRRT in finding optimal values of learning rate and weight decay for each encoder. This involved two separate runs of the LRRT. For the first run, we used a fixed value of weight decay (1×10^{-4}) and linearly increased the learning rate over 10 epochs. The results of the first run are given in Table 4.3. Additionally, Figure 4.5 shows the loss curves for each encoder as the learning rate is increased.

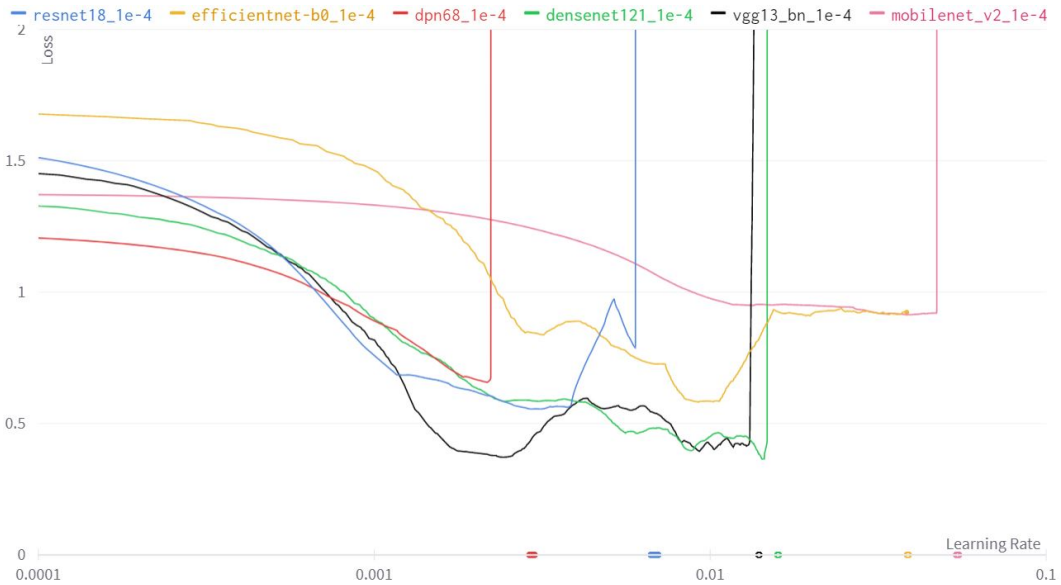


Figure 4.5: Baseline results for the LRRT.

It is discussed in [64] that although the LRRT can find the max LR, that value should not be used for training because it will lead to instability. Therefore, we

choose the optimal LR by selecting a point for each line in Figure 4.5 such that the loss has started to plateau.

Table 4.3: Baseline results for the LRRT.

Encoder	**Max LR**	**Optimal LR**
Mobilenet_v2	4.4×10^{-2}	3.5×10^{-2}
Efficientnet-b0	3.8×10^{-2}	1×10^{-2}
Dpn68	2.1×10^{-3}	1.5×10^{-3}
Resnet18	5.8×10^{-3}	3×10^{-3}
Densenet121	1.5×10^{-2}	1.1×10^{-2}
Vgg13_bn	1.4×10^{-2}	1×10^{-2}

Next, we find optimal values for weight decay. We perform a grid search with five possible values for weight decay. The values are 1×10^{-5}, 5×10^{-5}, 1×10^{-4}, 5×10^{-4}, 1×10^{-3}. Optimal learning rates from table 4.3 are used in each run. A full list of values for each encoder can be seen in table 4.4.

Surprisingly, weight decay had varying effects depending on the choice of encoder. Of the six encoders that were tested, two performed better with a larger value for weight decay, three with a smaller value, and one with the baseline value. We reserve the discussion as to why this might be the case until after an individual examination of each encoder. Let's start by examining the outlier, DenseNet121. Figure 4.6 shows that the baseline value for weight decay, 1×10^{-4}, results in the lowest overall loss value. While the run with a learning rate of 1×10^{-3} takes longer to explode, it shows signs of instability at higher learning rates. Therefore, the baseline learning rate is chosen for future experiments.

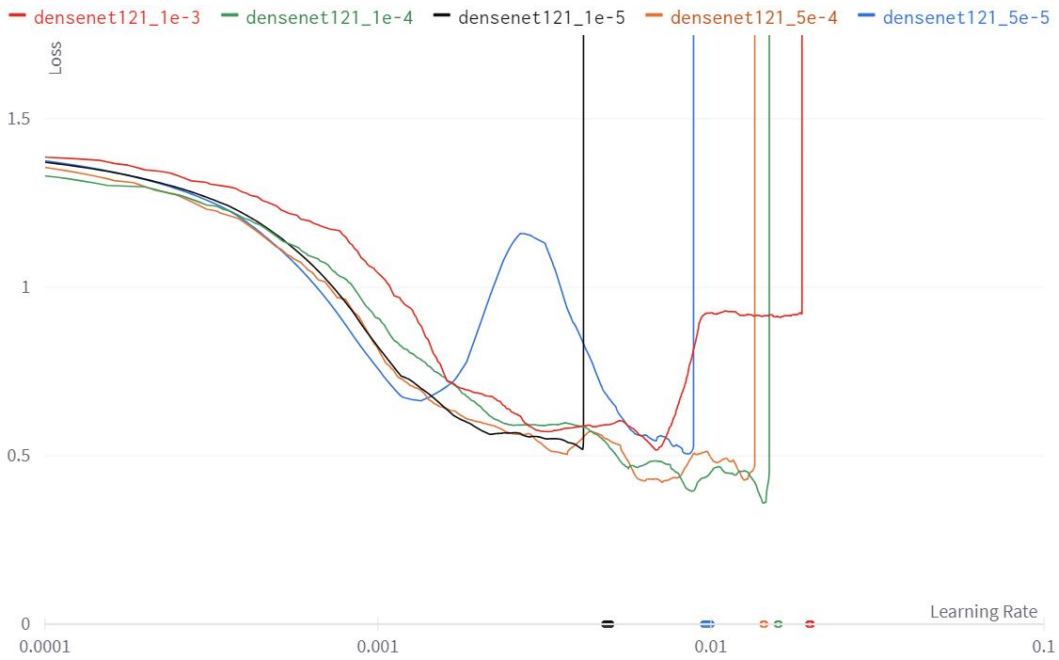


Figure 4.6: Weight decay LRRT for densenet121

Mobilenet_v2 (Fig 4.8), VGG_13bn (Fig 4.9) and EfficientNet-b0 (Fig 4.7) performed best by using a smaller weight decay value. The aforementioned encoders achieved the minimum loss when trained with a weight decay value of 1×10^{-5}. For EfficientNet-b0, the network performance deteriorates at large weight decay values, and does not converge. We can also notice that MobileNet_v2 is relatively stable for all tested values.

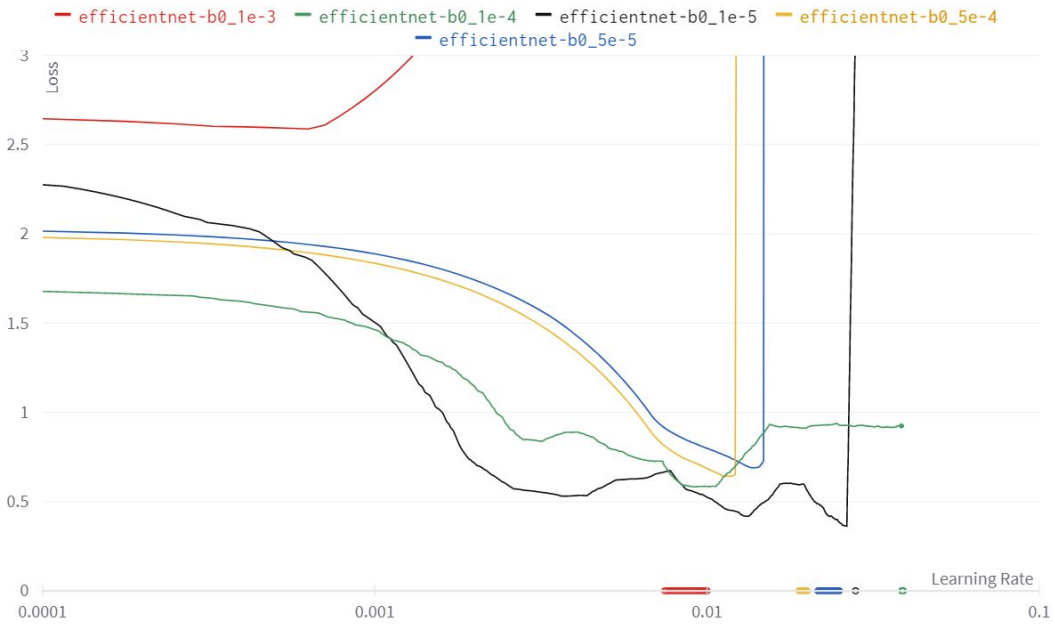


Figure 4.7: Weight decay LRRT for efficientnet-b0.

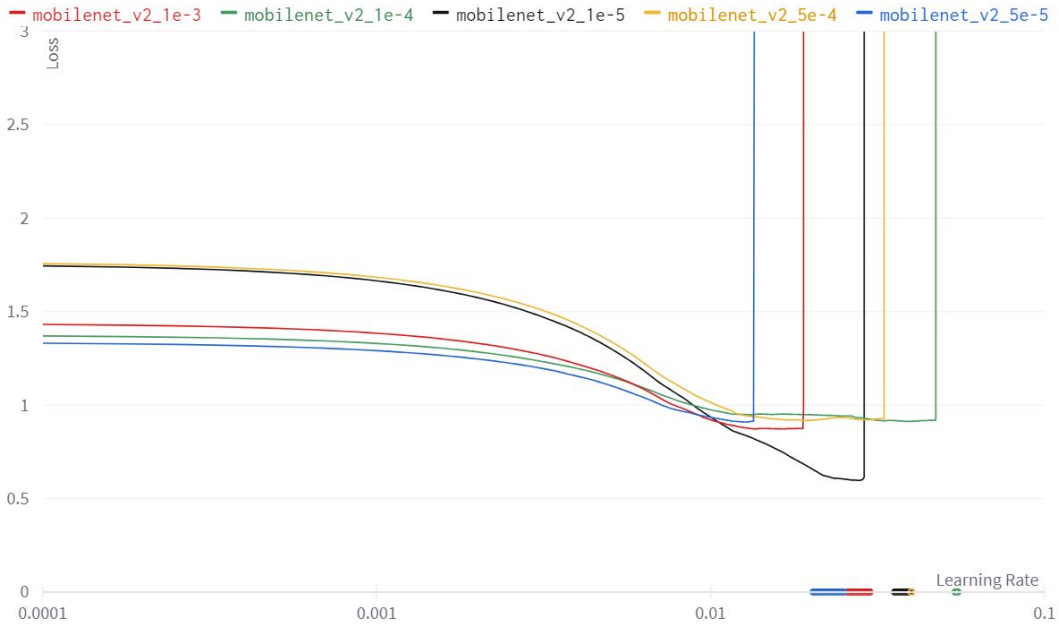


Figure 4.8: Weight decay LRRT for mobilenet_v2.

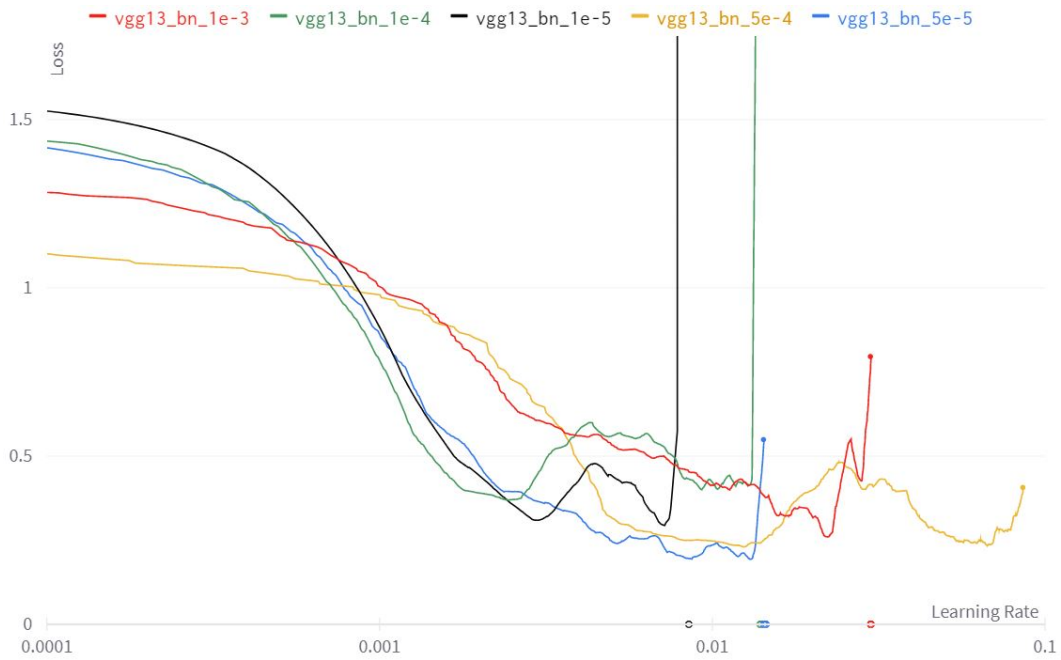


Figure 4.9: Weight decay LRRT for vgg13_bn.

Finally, we investigated if a large value of weight decay was beneficial. ResNet18 and DenseNet121 perform best with large weight decays, compared to the other networks. Most notably, ResNet18 does not experience exploding gradients for three of the runs. Further experiments are needed to investigate the cause of this anomaly.

Figure 4.10: Weight decay LRRT for rensenet18.

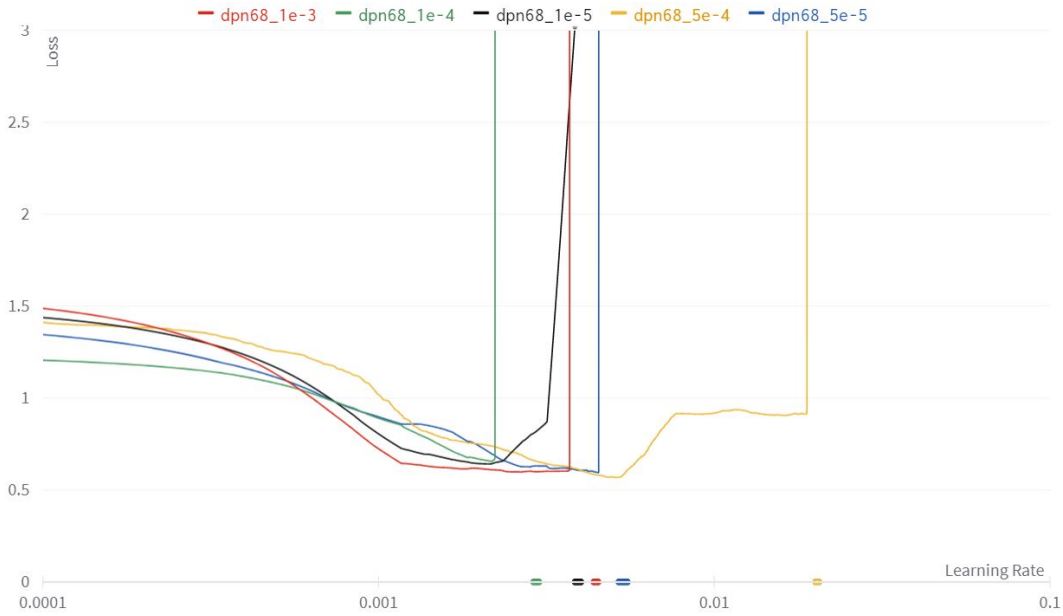


Figure 4.11: Weight decay LRRT for dpn68.

Table 4.4 summarises the training hyper-parameters that are found experimentally in this section. The values in this table will be used when comparing encoders in Section 4.5.4.

Table 4.4: Optimal learning rate and weight decay for each model.

Encoder	Optimal LR	Optimal WD
MobileNet_v2	3.5×10^{-2}	1×10^{-5}
EfficientNet-b0	1×10^{-2}	1×10^{-5}
DPN68	1.5×10^{-3}	5×10^{-4}
ResNet18	3×10^{-3}	1×10^{-3}
DenseNet121	1.1×10^{-2}	1×10^{-4}
VGG13_bn	1×10^{-2}	5×10^{-4}

4.5.2 Cyclic and OneCycle learninig rate schedules

We investigate if models trained with SGD and cyclic/OneCycleLR schedulers can outperform the popular optimisers that are discussed in Section 4.3. We select a U-net model with a MobileNet_v2 encoder for testing to minimise experiment run-time. Each model was trained for 100 epochs with a batch size of 24 for a range of learning rate values. For brevity, we will compare the best runs for each optimiser. Each model will be evaluated using the four metrics described in Section 4.4.

Our results indicate that the OneCycle learning rate with SGD achieved the best performance across the four metrics. Cyclic learning rate with SGD also showed a modest improvement over other optimiser methods. Our intuition is that the OneCycle learning rate facilitated the model to generalise and distinguish between similar classes, namely MLO and rock. As a result, the cool-down annihilation phase (see Figure 4.3) was initiated next to a local minima in the loss landscape.

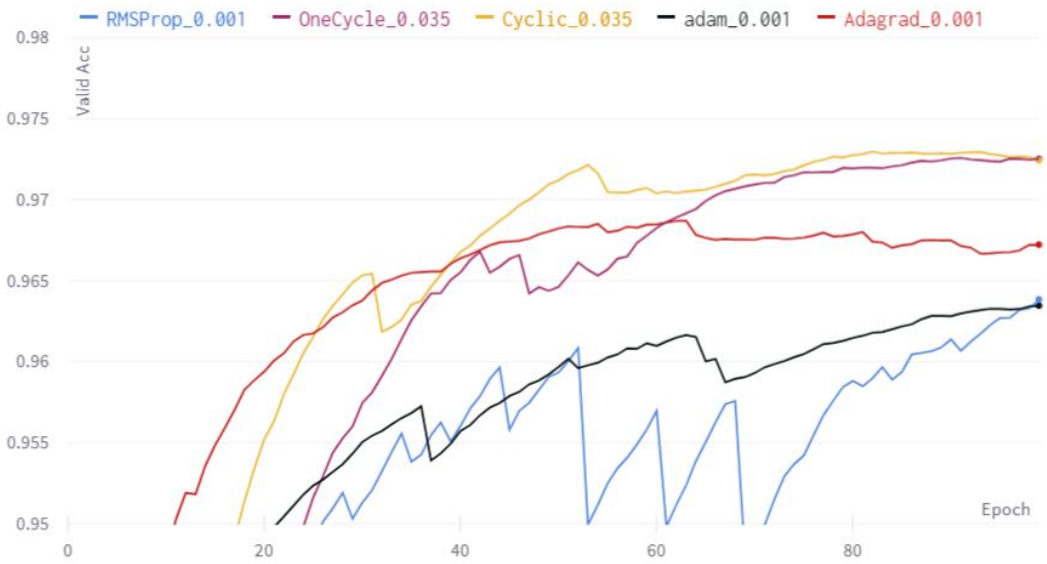


Figure 4.12: Optimiser and learning rate scheduler comparison: Accuracy.

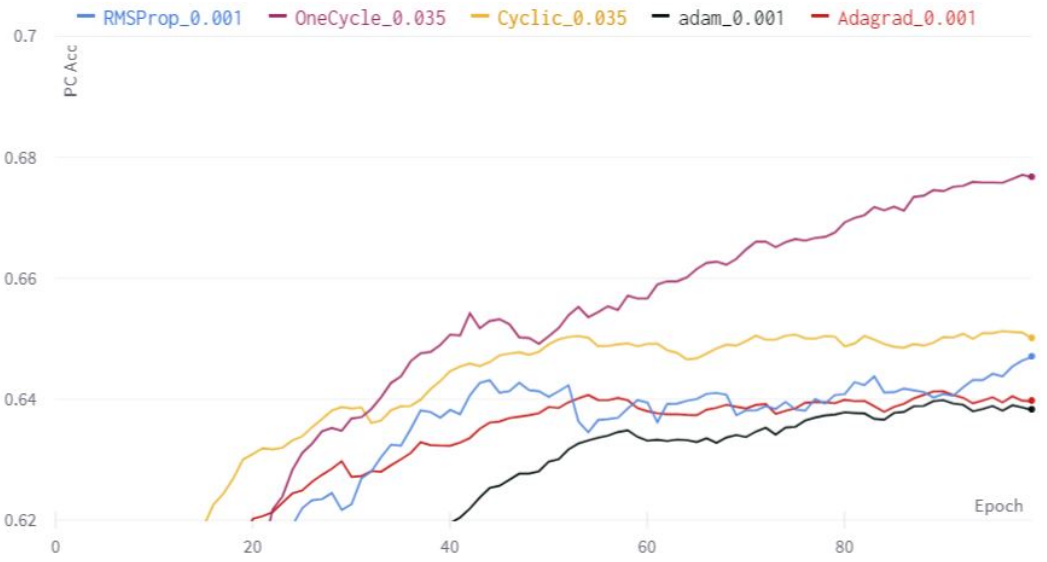


Figure 4.13: Optimiser and learning rate scheduler comparison: Mean accuracy.

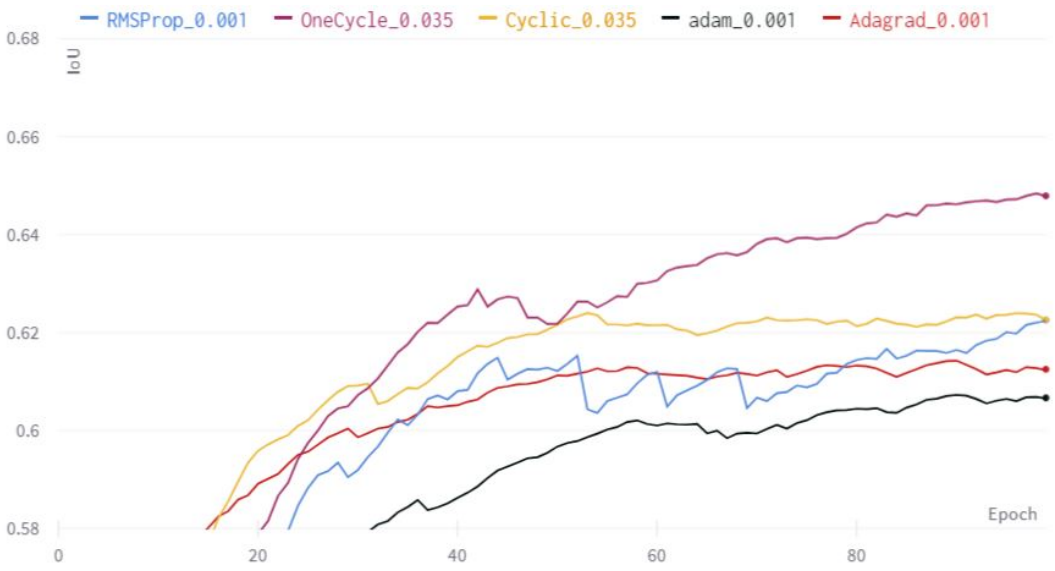


Figure 4.14: Optimiser and learning rate scheduler comparison: Mean IoU.

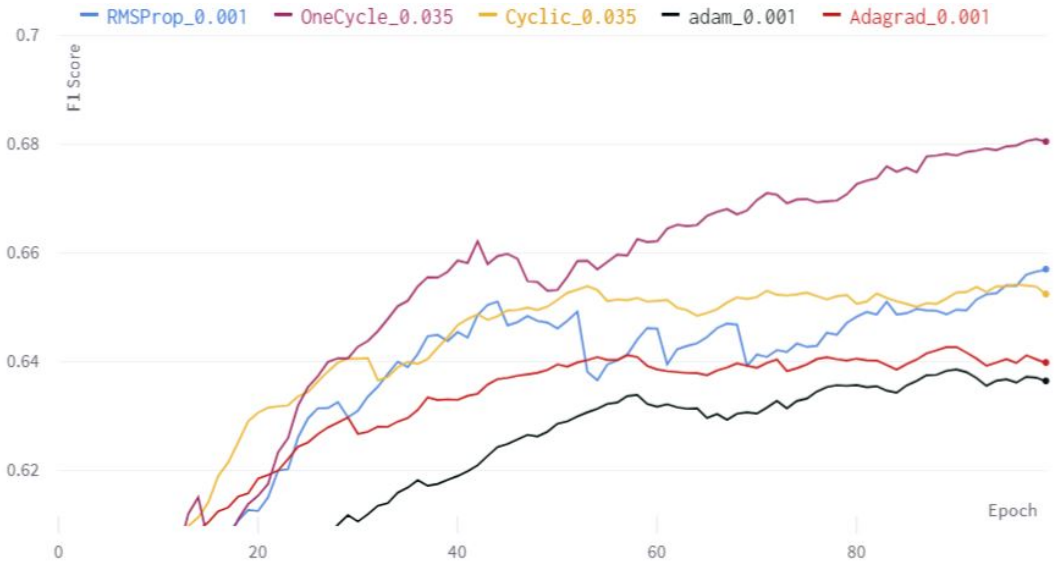


Figure 4.15: Optimiser and learning rate scheduler comparison: F1 score.

4.5.3 Median frequency balancing

In this section, we evaluate the MFB technique. A U-net model with a MobileNet_v2 encoder is used, similarly to Section 4.5.2. All other training parameters are also kept the same. Each configuration will be run three times. The weight vector is calculated according to Equation 4.9.

We obtain a weight vector, W, with values [1.06, 0.18, 12.16, 0.52, 438.55] given the class distribution shown in Table 4.2. Additionally, we modify the weight vector by reducing the large value which may lead to instability during training. We apply an arbitrary rule to scale down values greater than 10 by a factor of 2, and values greater than 100 by a factor of 10. This results in a weight vector with the following values - [1.06, 0.18, 6.08, 0.52, 43.855]. Each weight vector is run three times for 100 epochs each. The two weight vectors are compared to three runs without MFB.

First, we observe from Figure 4.16 that accuracy is the highest when no MFB is used. Interestingly, the runs with no MFB performed the worst across three other evaluation metrics. This indicates that the common classes, such as nadir, sand and sand ripple, are more accurately segmented when MFB is not applied. A possible explanation is that poor accuracy in the rare classes has little impact on the overall accuracy. For example, our model has 0% accuracy for the rock and MLO class, but it still produces a maximum accuracy of 99.12%.

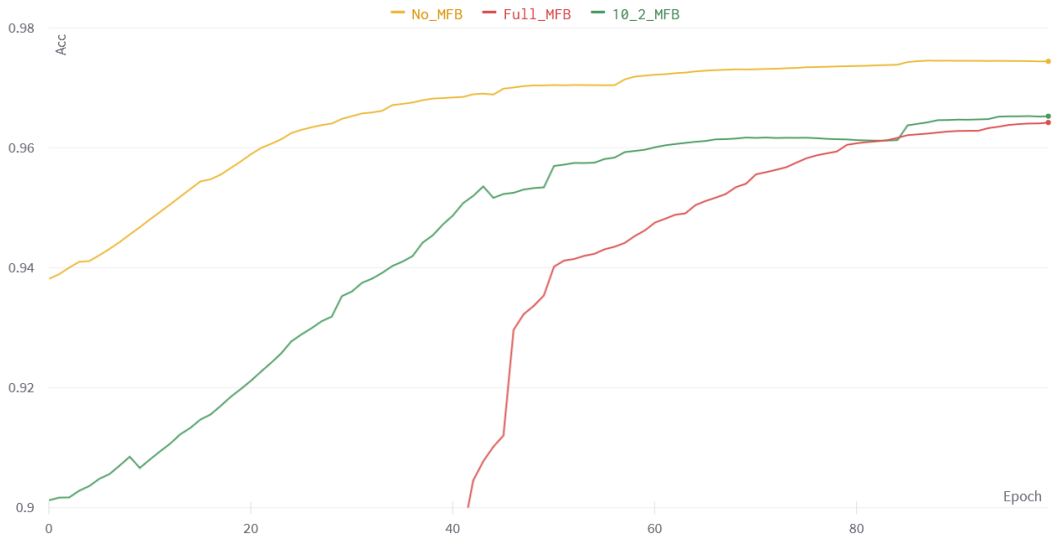


Figure 4.16: Median frequency balancing - accuracy.

Overall, we observed that MFB consistently improves the mean accuracy of our models. The full MFB and adjusted MFB models achieved on average a mean accuracy that was 25.1% and 27.3% higher, respectively. This is shown in Figure 4.17. Similarly, both the F1 score and mIOU metrics achieve their best scores when our scaled MFB technique is used, as shown in Figure 4.18 and Figure 4.19.

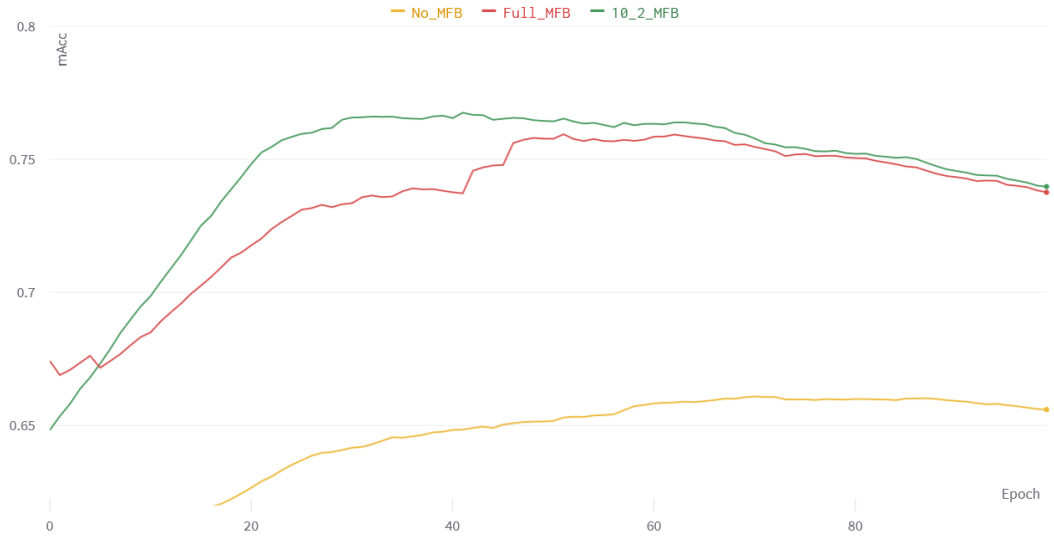


Figure 4.17: Median frequency balancing - mean accuracy.

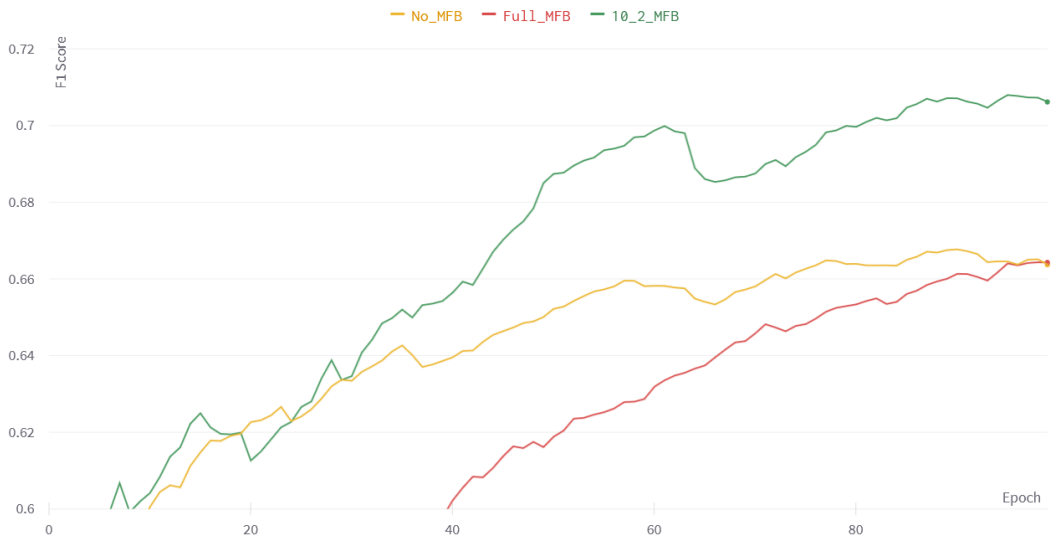


Figure 4.18: Median frequency balancing - F1 score.

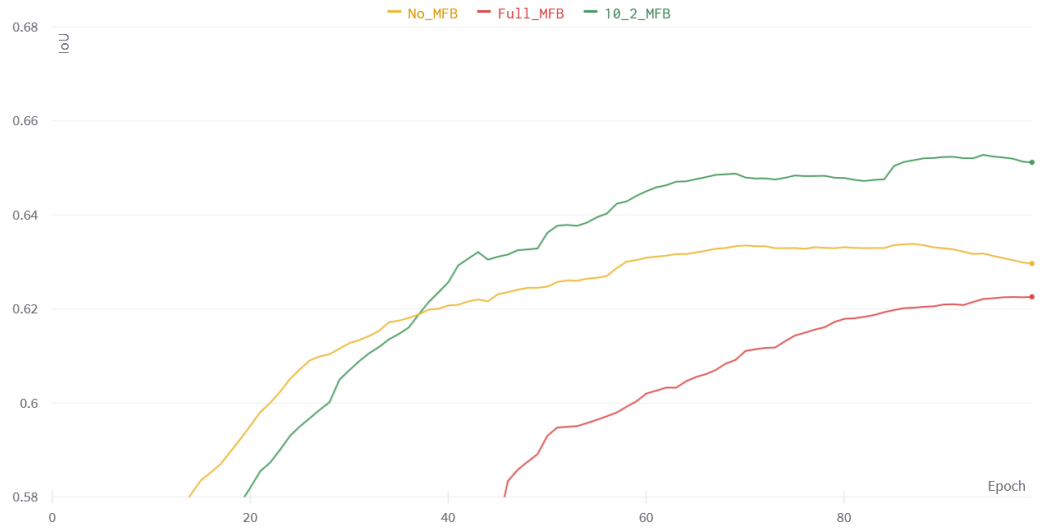


Figure 4.19: Median frequency balancing - mIoU.

4.5.4 Comparing encoders

In this section, we present the results from our encoder evaluations. We found EfficientNet-b0 to the best encoder in terms of segmentation performance. The model with the EfficientNet-b0 encoder performed the best in three metrics. The model using an EfficientNet-b0 backbone takes 15.8ms to process a batch of 16 image patches. This corresponds to 63 frames per second, which satisfies the requirement of real-time processing. It is also the second smallest model in terms

of parameters, which means it will require less resources from the deployed hardware.

It is interesting to note that all of the encoders performed comparably. We attribute the similarity to the hyper-parameter optimisation that was performed before this experiment. In Section 4.5.5, we will use the best performing encoder, EfficientNet-b0, to compare the performance of the selected segmentation architectures.

Table 4.5: The results of the encoder test.

Encoder	mAcc	F1	mIoU	Latency (ms)	Params (M)
Mobilenet_v2	0.767	0.738	0.685	10.2	**2.5**
Efficientnet-b0	**0.781**	**0.746**	**0.692**	15.8	4.8
Dpn68	0.764	0.729	0.678	21.3	11
Resnet18	0.761	0.730	0.676	6.1	11.2
Densenet121	0.763	0.734	0.684	24.4	6.2
Vgg13_bn	0.763	0.736	0.685	**5.5**	9

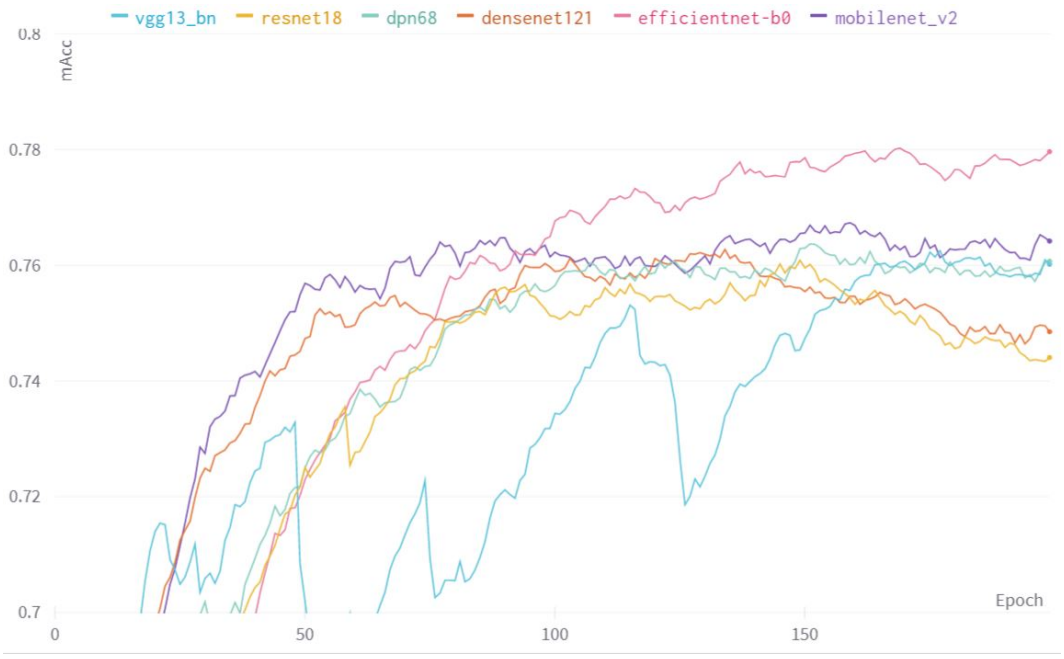


Figure 4.20: Encoder test mean accuracy.

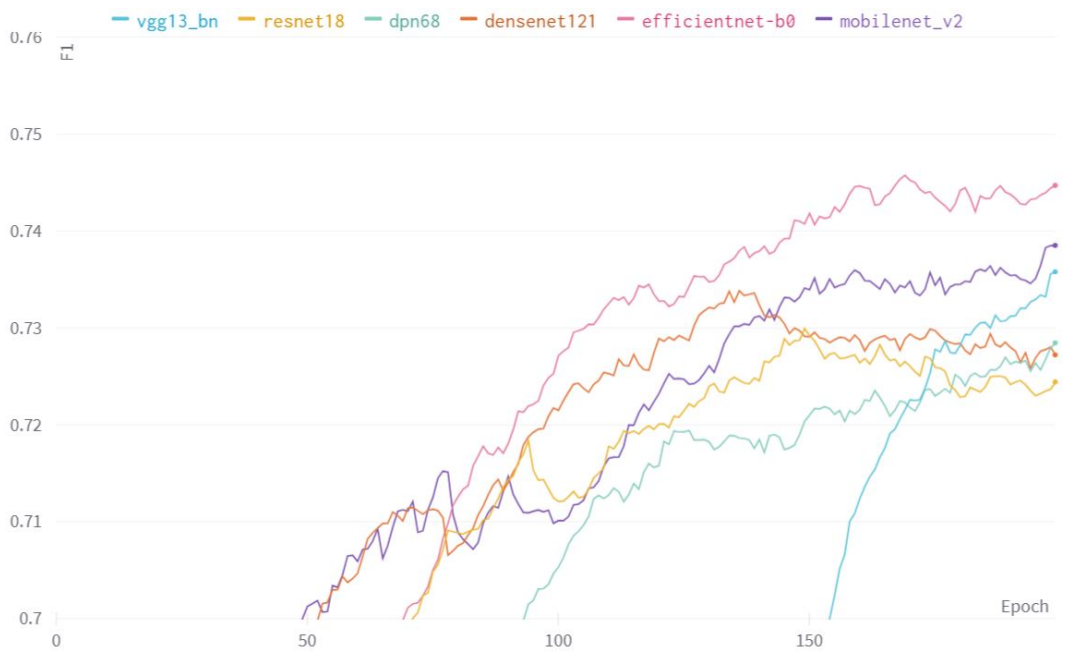


Figure 4.21: Encoder test F1 scores.

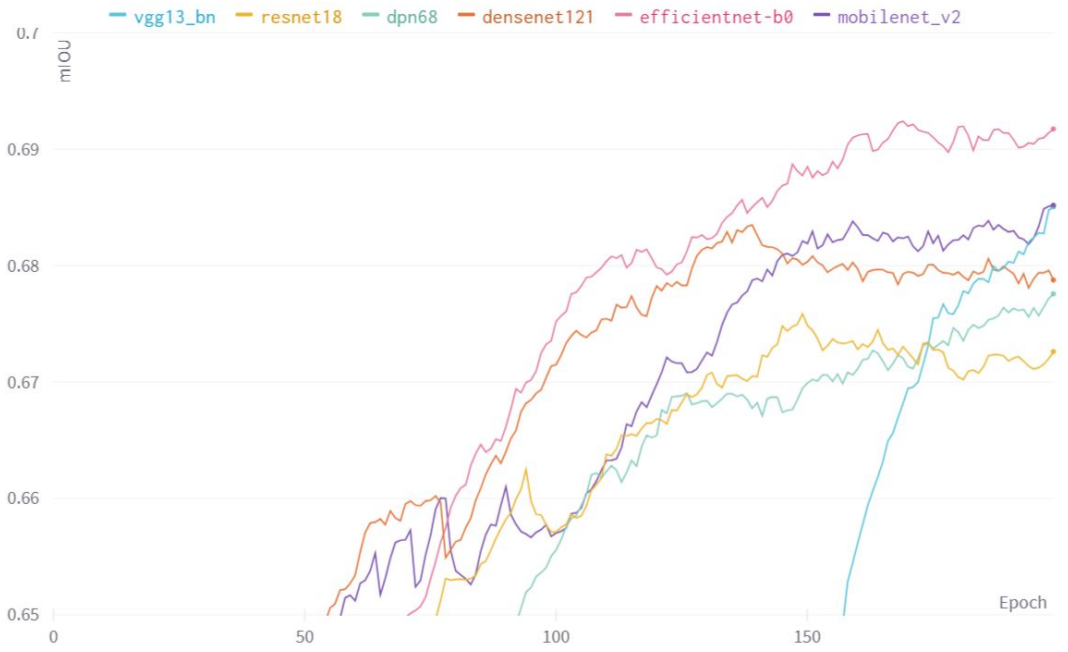


Figure 4.22: Encoder test mean IoU.

4.5.5 Comparing architectures

In this section, we evaluate seven segmentation architectures in combination with EfficientNet-b0, the best encoder from Section 4.5.4. The architectures are: U-Net [51], U-Net++ [75], DeepLabV3 [56], DeepLabV3+ [57], MA-Net [76], PAN [77] and FPN [53]. Table 4.6 gives a summary of our experimental results. Additionally, we plot the metrics during training in Figures 4.23 to 4.25.

We focus on U-Net++ and DeepLabV3 because they score the highest on the selected metrics. DeepLabV3 achieves a mean accuracy score that is 1.5% higher than U-Net++. However, U-Net++ has a similar improvement over DeepLabV3 in F1 score and mIoU, with an improvement of 1.9% in both metrics. The comparison is further complicated if we consider latency and model size. U-Net++ is 44.9% faster and 133% smaller. Despite this, both models operate at real-time speed, and have a small enough memory size to fit into embedded hardware.

Table 4.6: The results of the architecture comparison experiment.

Architecture	mAcc	F1	mIoU	Latency (ms)	Params (M)
U-Net	0.791	0.726	0.675	15.4	4.8
U-Net++	0.791	**0.735**	**0.683**	14.7	5.4
DeepLabV3	**0.803**	0.721	0.670	21.3	12.6
DeepLabV3+	0.777	0.709	0.662	**14**	4.3
MA-Net	0.785	0.725	0.676	16.2	6.1
PAN	0.776	0.702	0.660	15.5	5.8
FPN	0.781	0.710	0.670	14.3	**4.2**

It is difficult to determine what the best overall model is from the results in Table 4.6. U-Net++ is best in terms of F1 score and mIoU which are the two most popular metrics in semantic segmentation. On the other hand, DeepLabV3 has the highest mean pixel accuracy across all classes. We weight mean accuracy above the other two metrics and deem DeepLabV3 to be the best model for the following

reason. F1 score and IoU take into consideration false positives in the calculation. For the seabed segmentation application, some classes such as rock and MLO are under-represented. Hence, the segmentation model is penalised heavily when predicted regions are larger than ground-truth regions, as in Example 4 and 5 of Figure 4.26. Predictions of MLO or rock regions that go beyond the border of the ground-truth cause a large decrease in mIoU and F1, whereas the same cannot be said for mAcc.

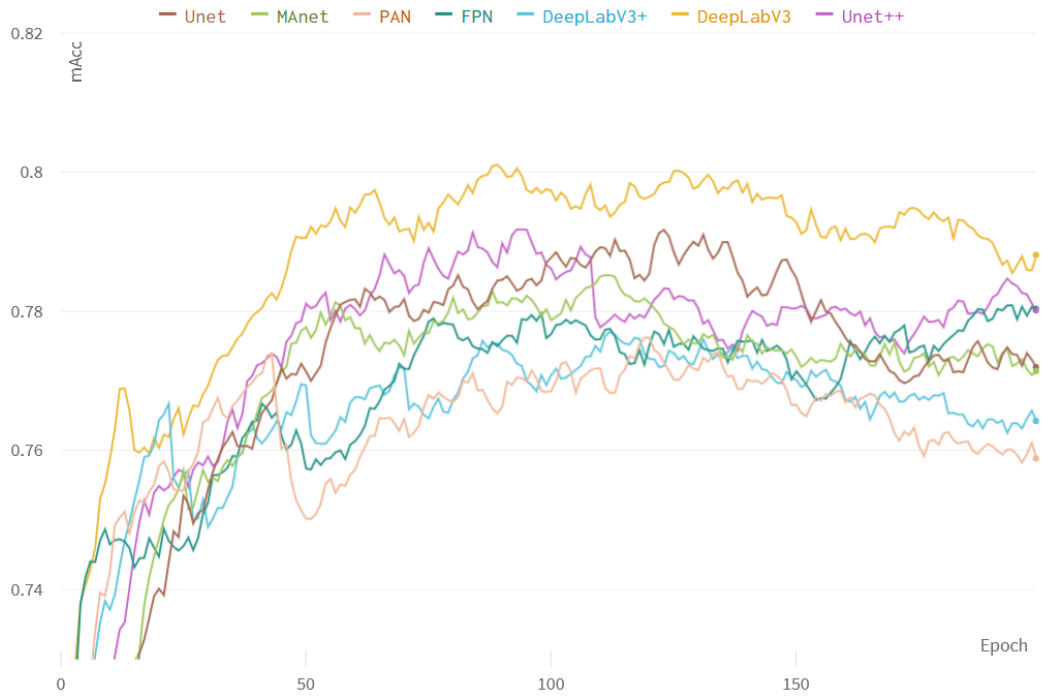


Figure 4.23: Architecture test mean accuracy.

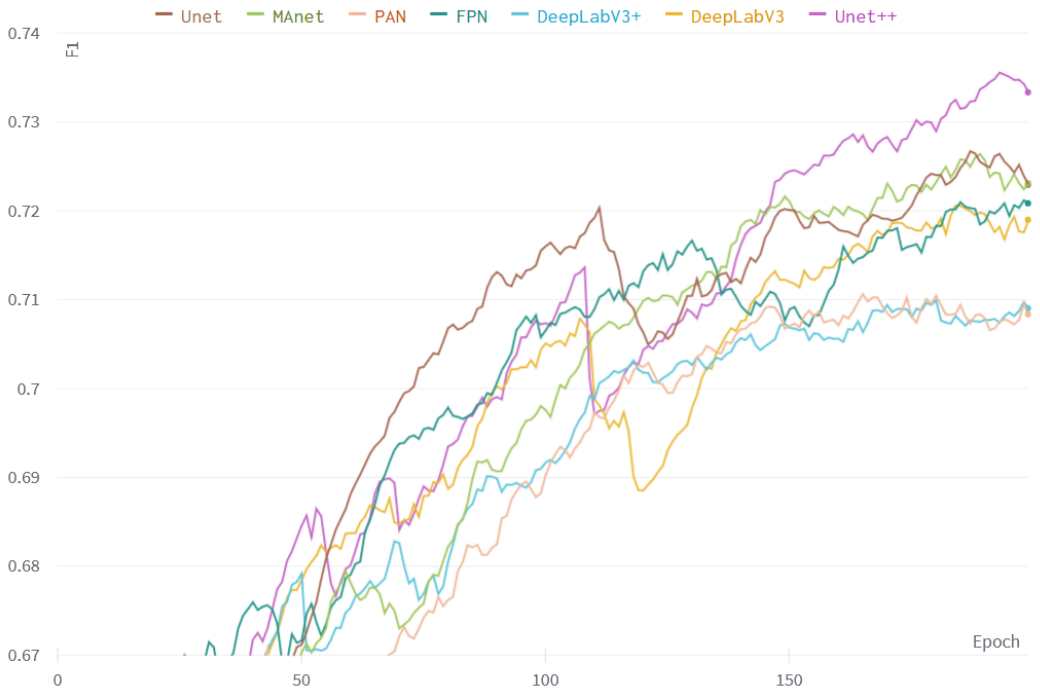


Figure 4.24: Architecture test F1 scores.

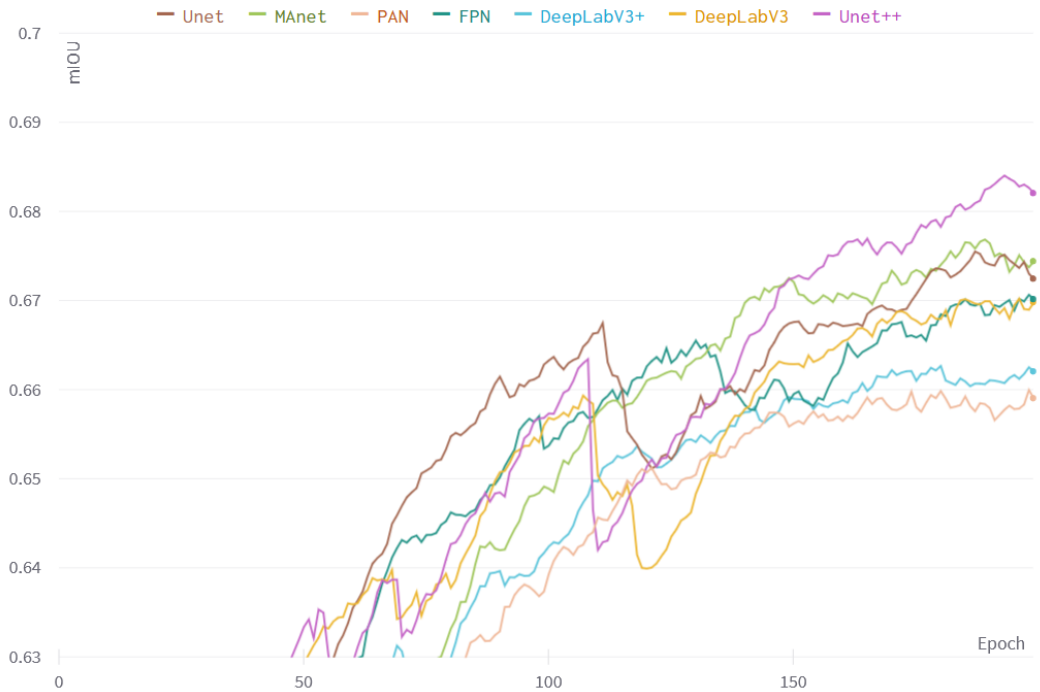


Figure 4.25: Architecture test mean IoU.

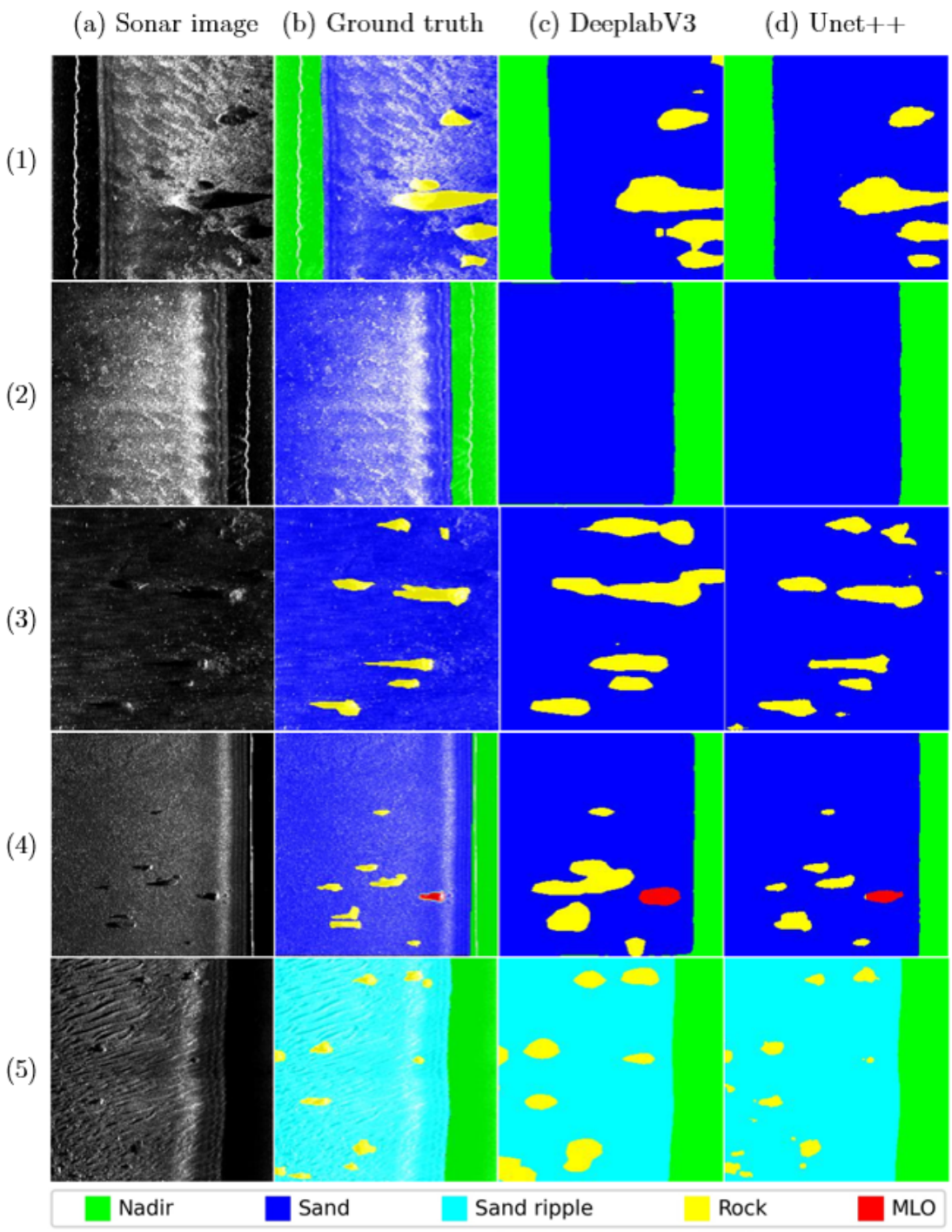


Figure 4.26: A visual comparison of the outputs from the two best segmentation models.

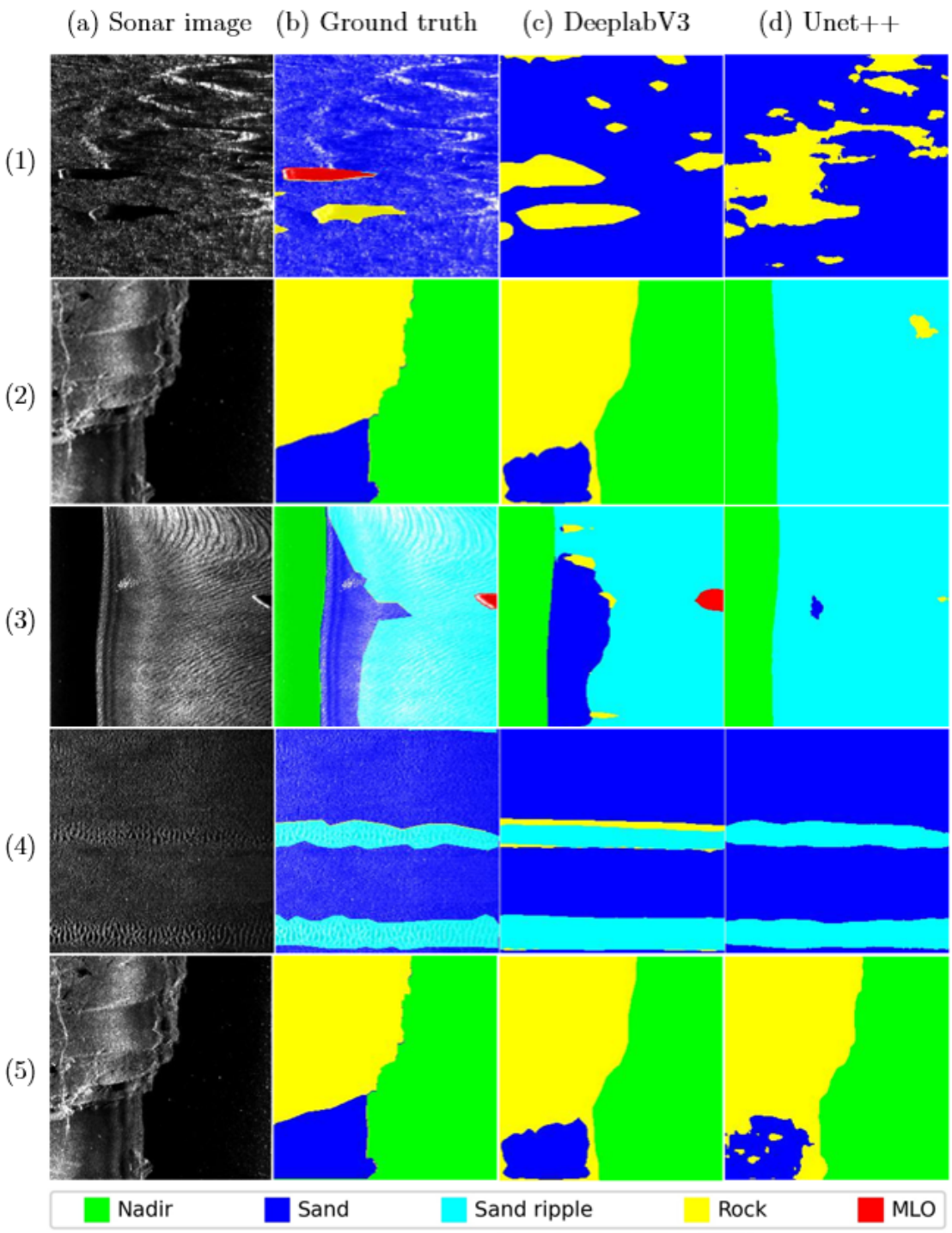


Figure 4.27: A visual comparison of the outputs from the two best segmentation models cont.

Chapter 5

Conclusion

Chapter contents

5.1 Summary of work 78

5.2 Future research directions 79

5.1 Summary of work

In this project, we have investigated deep learning algorithms for two important tasks in undersea surveillance: i) naval mine detection; and ii) seabed terrain segmentation.

Our experiments indicate that deep learning is a promising approach in both naval mine detection and seabed terrain segmentation. General purpose object detection models (e.g. YOLO, EfficientDet, and R-CNN) were found to be applicable to sonar images. Furthermore, models that were pretrained on the ImageNet dataset were better at detecting mines than models trained with randomly initialised weights across all metrics. We found that data augmentation techniques consistently improved the performance of the naval detection models. Additionally, our ECP algorithm provided a means to generate additional labelled data from our large database of unlabelled data. The use of ECP led to a slight improvement in performance. Data augmentation only incurs a computational cost

at training time which solidifies its usefulness in real-time applications.

Similar conclusions can be drawn for seabed segmentation. We found that general purpose segmentation models (e.g. MobileNet, EfficientNet, and DenseNet) can be readily applied to sonar data. Our experimentation showed that median frequency balancing mitigated the negative effect of imbalance in the representation of classes throughout the dataset. Comparing model configuration was a major focus in our experiment. Here we applied various methods such as the learning rate range test and weight decay range test to find optimal model training configurations. We established that alternate learning rate schedules (e.g. OneCycle and Cyclic) minimised overfitting while training on small datasets.

5.2 Future research directions

There are several promising directions for future research related to this project. The first direction is to investigate recent object detection algorithms (e.g. YOLOv5, YOLOv7, YOLOvX, and Swin Transformers) and apply them for naval mine detection. The second direction is to investigate recent semantic segmentation algorithms (e.g. HRNet-OCR and HSMA) and apply them for seabed terrain segmentation. The third direction is to develop few-shot machine learning algorithms to train a mine detection model using only a small number of labelled images. The fourth direction is to investigate data augmentation techniques (e.g. RandAugment and AutoAugment) and apply them to both our detection and segmentation datasets.

Bibliography

[1] J. Avery, *The Naval Mine Threat to U.S. Surface Forces*. Arlington, VA: Surface Warfare, 1998.

[2] National Research Council, *Oceanography and Mine Warfare*. Washington, DC: The National Academies Press, 2000.

[3] J. M. Boorda, "Mine countermeasures - an integral part of our strategy and our forces," 1999. [Online]. Available: https://fas.org/man/dod-101/sys/ship/weaps/docs/cnopaper.htm

[4] S. G. Johnson and M. A. Deaett, "The application of automated recognition techniques to side-scan sonar imagery," *IEEE Journal of Oceanic Engineering*, vol. 19, pp. 138–144, 1994.

[5] S. B. Morien, *The Operational Effects of Mine Warfare*. Newport, RI: Naval War College, 1999.

[6] *Delph Sonar Advanced Notes*, 1st ed. Boston, Massachusetts: iXBlue, 2013.

[7] J. McKay, I. Gerg, V. Monga, and R. Raj, "What's mine is yours: Pretrained cnns for limited training sonar atr," in *OCEANS*, 2017, pp. 1–7.

[8] P. Chapple, T. Dell, and D. Bongiorno, "Enhanced detection and classification of mine-like objects using situational awareness and deep learning," in *UACE Proceedings*, 2017, pp. 529–536.

[9] D. Einsidler, M. Dhanak, and P. P. Beaujean, "A deep learning approach to target recognition in side-scan sonar imagery," in *OCEANS*, 2018, pp. 1–4.

[10] A. Bouzerdoum, P. B. Chapple, M. Dras, Y. Guo, L. Hamey, T. Hassanzadeh, T. Le Hoang, O. M. Nazmi, M. Orgun, S. L. Phung, C. Ritz, and M. Shahpasand, "Improved deep-learning-based classification of mine-like contacts in sonar images from autonomous underwater vehicles," in *Underwater Acoustic Conference*, 2019, pp. 179 – 186.

[11] G. Huo, Z. Wu, and J. Li, "Underwater object classification in sidescan sonar images using deep transfer learning and semisynthetic training data," *IEEE Access*, vol. 8, pp. 47 407–47 418, 2020.

[12] H. T. Le, S. L. Phung, P. B. Chapple, A. Bouzerdoum, C. H. Ritz, and L. C. Tran, "Deep gabor neural network for automatic detection of mine-like objects in sonar imagery," *IEEE Access*, vol. 8, pp. 94 126–94 139, 2020.

[13] S. L. Phung, T. N. A. Nguyen, H. T. Le, P. B. Chapple, C. H. Ritz, A. Bouzerdoum, and L. C. Tran, "Mine-like object sensing in sonar imagery with a compact deep learning architecture for scarce data," in *Digital Image Computing: Techniques and Applications*, 2019.

[14] D. G. Tucker, "Sonar and underwater acoustical engineering," *Electronics and Power*, vol. 11, no. 7, pp. 220–223, 1965.

[15] A. Wood, F. Smith, and J. McGeachy, "A magnetostriction echo depth-recorder," *Proceedings of the Wireless Section of the Institution*, vol. 10, no. 29, pp. 80–93, 1935.

[16] M. W. Atherton, *Echoes and Images*. Vancouver, BC: OysterInk Publications, 2011.

[17] H. U. Schnitzler and E. K. V. Kalko, "Echolocation by insect-eating bats," *BioScience*, vol. 51, no. 7, pp. 557–569, 2001.

[18] M. Johnson, P. T. Madsen, W. M. X. Zimmer, N. Aguilar De Soto, and P. L. Tyack, "Beaked whales echolocate on prey," *Proceedings of the Royal Society B*, vol. 271, no. 6, pp. 383–386, 2004.

[19] C. M. Ciany and J. Huang, "Computer aided detection/computer aided classification and data fusion algorithms for automated detection and classification of underwater mines," in *OCEANS*, vol. 1, 2000, pp. 277–284.

[20] G. J. D. Dobeck, J. C. H. Hyland, and L. D. Smedley, "Automated detection and classification of sea mines in sonar imagery," in *Proceedings of SPIE*, vol. 3079, 1997.

[21] D. M. Lane and J. P. Stoner, "Automatic interpretation of sonar imagery using qualitative feature matching," *IEEE Journal of Oceanic Engineering*, vol. 19, no. 3, pp. 391–405, 1994.

[22] Y. C. Chang, S. K. Hsu, and C. H. Tsai, "Sidescan sonar image processing: Correcting brightness variation and patching gaps," *Journal of Marine Science and Technology*, vol. 18, 2010.

[23] D. P. Williams, V. Myers, and M. S. Silvious, "Mine classification with imbalanced data," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 3, pp. 528–532, 2009.

[24] M. A. Pinto, A. Bellettini, R. Hollett, and A. Tesei, "Real- and synthetic-array signal processing of buried targets," *IEEE Journal of Oceanic Engineering*, vol. 27, pp. 484–494, 2002.

[25] A. Nevis, J. Bryan, J. Taylor, and B. Cordes, "A baseline object detection using a background anomaly approach for electro-optic identification sensors," *OCEANS*, pp. 1–9, 2002.

[26] M. P. Strand, "Computer aided detection and classification of mines using a fluorescence imaging laser line scan (fills) sensor," *International Conference*

on Requirements & Technologies for the Detection, Removal & Neutralization of Landmines & UXO, pp. 1–11, 2003.

[27] G. Russell and D. Lane, "A knowledge-based system framework for environmental perception in a subsea robotics context," *IEEE Journal of Oceanic Engineering*, vol. 11, pp. 401–412, 1986.

[28] S. Reed, Y. Petillot, and J. Bell, "An automatic approach to the detection and extraction of mine features in sidescan sonar," *IEEE Journal of Oceanic Engineering*, vol. 28, no. 1, pp. 90–105, 2003.

[29] Y. Petillot, J. Bell, and S. Reed, "Automated approach to classification of mine-like objects in sidescan sonar using highlight and shadow information," *IEEE Proceedings - Radar, Sonar and Navigation*, vol. 151, no. 1, pp. 48–56, 2004.

[30] D. P. Williams, V. Myers, and M. S. Silvious, "Mine classification with imbalanced data," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 3, pp. 528–532, 2009.

[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, p. 211–252, 2015.

[35] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[36] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," *Advances in Neural Information Processing Systems*, vol. 2, p. 2553–2561, 2013.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 2, 2012, pp. 1097–1105.

[38] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations*, 2013, pp. 1–16.

[39] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[40] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6517–6525.

[41] ——, "Yolov3: An incremental improvement," *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–6, 2018.

[42] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[43] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[44] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of Machine Learning Research*, vol. 97, 2019, pp. 6105–6114.

[45] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 2013.

[46] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[47] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 580–587.

[48] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Conference on Neural Information Processing Systems*, vol. 1, 2015, p. 91–99.

[49] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, 2015.

[50] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[51] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

[52] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125.

[53] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6399–6408.

[54] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis Machine Intelligence*, no. 4, pp. 834–848, 2018.

[55] ——, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[56] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *European Conference on Computer Vision (ECCV)*, pp. 883–851, 2018.

[57] ——, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision*, pp. 801–818.

[58] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[59] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," *IEEE International Conference on Computer Vision*, pp. 1310–1319, 2017.

[60] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," *IEEE Conference on Computer Vision and Patter Recognition*, pp. 2315–2324, 2016.

[61] R. Takahashi, T. Matsubara, and K. Uehara, "Data augmentation using random image cropping and patching for deep CNNs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 2917 – 2931, 2019.

[62] PythonLessons, "Tensorflow-2.x-yolov3," 2020, GitHub Repository. [Online]. Available: https://github.com/pythonlessons/TensorFlow-2.x-YOLOv3

[63] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7310–7319.

[64] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006. International Society for Optics and Photonics, 2017, p. 1100612.

[65] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision*. IEEE, pp. 464–472.

[66] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 770–778.

[68] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[69] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, pp. 1–14.

[70] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, pp. 1–13, 2015.

[72] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[73] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[74] P. Yakubovskiy, "Segmentation models pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2020.

[75] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "U-Net++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer International Publishing, pp. 3–11.

[76] T. Fan, G. Wang, Y. Li, and H. Wang, "MA-Net: A multi-scale attention network for liver and tumor segmentation," *IEEE Access*, vol. 8, pp. 179 656–179 665, 2020.

[77] H. Li, P. Xiong, J. An, and L. Wang, "Pyramid attention network for semantic segmentation," *British Machine Vision Conference (BMVC)*, pp. 1–13, 2018.