Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Jonas Nagell Borgersen

Attention segmentation approaches for plankton images captured in-situ

Master's thesis in Cybernetics and Robotics
Supervisor: Annette Stahl
Co-supervisor: Aya Saad
June 2021

NTNU
Norwegian University of
Science and Technology

Jonas Nagell Borgersen

Attention segmentation approaches for plankton images captured in-situ

Master's thesis in Cybernetics and Robotics
Supervisor: Annette Stahl
Co-supervisor: Aya Saad
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

Insight on changes in abundance and distribution of plankton can help scientist better understand effects of climate change on marine ecosystems. The aim of this thesis is to find efficient methods for attention segmentation for real-time detection of plankton in images captured by an autonomous underwater vehicle (AUV). Attention segmentation refers to pixel-wise segmentation of foreground from background. In recent years, Deep convolutional neural networks have conquered most computer vision fields. For the computer vision task of attention segmentation, convolutional neural networks with encoder-decoder structures have been proven successful.

In this thesis, comprehensive testing of deep learning methods for attention segmentation is carried out. Five networks for attention segmentation are implemented and tested. These are U-net, U-net++, LinkNet, DeepLab V3+ and Pyramid attention network (PAN). Several loss functions are tested in training of networks to find which loss functions are best suited for the segmentation task. As backbones for the networks for attention segmentation, five pre-trained encoders are implemented and tested. Through a process of elimination, candidate methods are narrowed down. The methods are measured on segmentation performance and run-time for predictions. Finally two methods are recommended for attention segmentation for the planktonic images. The first method is the method achieving the best segmentation performance in experiments, and the second method is a method yielding low run-times and high performance in experiments.

The planktonic dataset for which attention segmentation is performed in this thesis is rather small, with only 312 manually labeled images. To generate additional training data, Mixture-Gaussian-based segmentation is carried out on images in the training set. This acts a special case of data augmentation. Segmentation performance is compared between models trained on the original dataset containing only manually labeled images, and models trained on an enlarged dataset containing manually labeled images and images labelled through Mixture-Gaussian-based segmentation. A small improvement in segmentation performance is obtained from training models on the enlarged dataset.

Sammendrag

Plankton er svært viktig for marine økosystemer, og for å forstå hvordan klimaendringer påvirker liv i vann er det nyttig å få innblikk i hvordan plankton bestander påvirkes av et endret klima. Målet med denne oppgaven er å finne effektive metoder for deteksjon av plankton i sanntid ved å skille forgrunn fra bakgrunn i bilder tatt av et autonomt undervannskjøretøy (AUV). Vi skiller forgrunn fra bakgrunn pixel for pixel, og dette kaller vi forgrunn-segmentering. I senere år har bruken av dype nevrale nett ført til store framskritt innenfor datasyn. I oppgaver som har til hensikt å klassifisere alle pixeler i bilder, har dype nevrale nettverk med enkoder-dekoder struktur vist seg å være effektive.

I denne oppgaven utføres omfattende testing av dype nevrale nettverk for forgrunn-segmentering. Fem nettverk for forgrunn-segmentering er implementert og testet. Disse er U-net, U-net ++, LinkNet, DeepLab V3 + og Pyramid attention network (PAN). Flere kost-funksjoner er testet i trening av nettverk for å finne hvilken kost-funksjon som er best egnet for formålet. Flere forhåndstrente enkodere testes sammen med nettverkene. Gjennom en elimineringsprosess siles metoder ut. Metodene vurderes etter hvor gode de er til å skille forgrunn fra bakgrunn, og etter hvor lang kjøretid de har for prediksjoner. Til slutt anbefales to metoder. Den første metoden er metoden som oppnår den beste ytelsen i eksperimenter, og den andre metoden er en metode som gir lave kjøretider og høy ytelse i eksperimenter.

Datasettet som brukes til forgrunn-segmentering er ganske lite, med bare 312 merkede bilder. Bildene er manuelt merket av en biolog. For å generere mer treningsdata, blir segmentering ved bruk av blandede gaussiske distribusjoner utført for bilder i treningssettet. Dette fungerer som et spesielt tilfelle av dataøkning. Ytelse sammenlignes mellom modeller som er trent på det originale datasettet som bare inneholder manuelt merkede bilder, og modeller som er trent på et forstørret datasett som inneholder manuelt merkede bilder og bilder som er merket gjennom segmentering med blandede gaussiske distribusjoner. En liten forbedring i ytelse oppnås fra modeller trent med det forstørrede datasettet.

Preface

This master thesis, TTK4900, is carried out in the department of Engineering cybernetics at the Norwegian University of Science and Technology.

I would like to thank my supervisors Associate Professor Annette Stahl and Postdocoral Fellow Aya Saad for valuable ideas and guidance during the writing of this thesis and for encouraging me to write a paper based on this work. The paper is accepted to be listed in conference proceedings for "the 14th International Conference on Machine Vision (ICMV 2021)". Furthermore, an extension of the work presented in the conference paper will be submitted for the "JAIT journal-Journal of Advances in Information Technology".

I also want to thank the Norwegian educational system for providing free education for me and all other Norwegians. Finally, I want to thank my friends for keeping spirits up at the office with our daily lunch-breaks.

Abbreviations

- CNN = Convolutional neural network
- PAN = Pyramid attention network
- BCE = Binary cross entropy loss
- BCE 2 = Binary cross entropy loss with positive weights set to 2
- BCE 7 = Binary cross entropy loss with positive weights set to 7
- BCE 15 = Binary cross entropy loss with positive weights set to 15
- Focal g1 = Focal loss with γ set to 1 and α set to 1
- Focal g2 = Focal loss with γ set to 2 and α set to 1
- Focal g3 = Focal loss with γ set to 3 and α set to 1

Contents

Abstract . . . iii
Sammendrag . . . v
Preface . . . vii
Abbreviations . . . ix
Contents . . . xi
1 Introduction . . . 1
1.1 Problem description . . . 1
1.2 Research questions . . . 2
1.3 Motivation . . . 3
1.4 Contribution . . . 4
1.5 Outline . . . 4
2 Theoretical background . . . 5
2.1 Image segmentation and attention segmentation . . . 5
2.2 Machine Learning . . . 6
2.3 Deep learning . . . 7
2.3.1 General artificial neural network architecture . . . 8
2.3.2 Feedforward neural networks . . . 8
2.3.3 Activation functions . . . 9
2.3.4 Loss functions . . . 10
2.3.5 Backpropagation and Optimizers . . . 11
2.3.6 Generalization over-fitting, and under-fitting . . . 11
2.3.7 Data augmentation in computer vision . . . 12
2.3.8 Convolutional neural networks . . . 13
3 Methods . . . 15
3.1 Deep learning models for attention segmentation . . . 15
3.1.1 U-net . . . 15
3.1.2 U-net++ . . . 16
3.1.3 Deeplab V3+ . . . 16
3.1.4 Pyramid attention network (PAN) . . . 18
3.1.5 LinkNet . . . 18
3.2 Pretrained encoders . . . 19
3.2.1 Deep residual networks (ResNet) . . . 20
3.2.2 EfficientNet . . . 21
3.2.3 MobileNet v2 . . . 22

3.3 Loss functions 23
3.4 Evaluation metrics 24
3.5 Mixture-Gaussian-based segmentation (MOG2) 25
3.6 Morphological operations 26
4 Datasets 27
4.1 Original planktonic image dataset 27
4.2 Planktonic image dataset enlarged with MOG2 labeled data 28
5 Framework and implementation 31
5.1 Pre-processing 31
5.2 Deep learning model (DL-Model) 32
5.3 Model training 32
5.3.1 Training data 32
5.3.2 hyper-parameters 32
5.4 Post-processing 33
5.5 Hardware specifications 33
5.6 Implementation details for Mixture-Gaussian-based segmentation . 33
6 Experiments 35
6.1 Deep learning attention segmentation with manually labeled data . 35
6.2 Training deep learning models on an enlarged dataset 37
7 Results 39
7.1 Training networks on the original dataset 39
7.1.1 Finding a suitable loss function 39
7.1.2 Testing all networks with ResNet-101 encoder 40
7.1.3 Testing U-net++ and LinkNet with different encoders 40
7.1.4 Testing U-net++ and LinkNet with selected encoders over an increased number of epochs 41
7.2 Training networks on the enlarged dataset 42
8 Discussion 49
8.1 The best performance achieved 49
8.2 The best deep learning model for real-time attention segmentation 49
8.3 The suitable loss function for the segmentation task 50
8.4 The effect of adding MOG to the training process 50
9 Conclusion 53
10 Future work 55
Bibliography 57
Appendix 63

Chapter 1

Introduction

Tracking changes in abundance and distribution of plankton can help scientists better understand the effect of climate change on marine life, as plankton are a vital foundation in marine ecosystems.

Phytoplankton, plankton that produce energy from photosynthesis, are responsible for half of global net primary production [1]. Primary production refers to the production of chemical energy from sunlight. Zooplankton, on the other hand, transfer energy from primary producers to other aquatic animals. They are grazers, that primarily eat phytoplankton, and in turn are eaten by larger animals [2]. The most common zooplankton, copepods, are the most abundant multicellular animals on Earth [2]. Since plankton are situated at the bottom of marine food chains, change in abundance and distribution of plankton will severely affect aquatic ecosystems [3]. A changing climate affects plankton. For instance, there has been a decline in the abundance of krill, a zooplankton that is a key component in the diets of whales and other mammals. This is linked with reduced food availability of phytoplankton in Antarctica caused by warmer waters [3]. Being free floaters, regional plankton distributions are sensitive to changing currents [3]. Furthermore warmer water in the upper water column affects vertical mixing of nutrition, which affects the growth of phytoplankton [1]. Gaining further understanding of abundance and distribution of plankton can provide important insight on the effect of climate change on life in the oceans [3].

1.1 Problem description

The work in this thesis aims at performing in-situ detection of planktonic data captured by a mobile platform, lightweight autonomous vehicle (AUV), described in [4]. The AUV captures images in the upper water column to map distributions of planktonic organisms. Image processing, i.e detection and classification of planktonic organisms, is done in-situ. This is made possible by advances in machine learning and real-time robotic visual sensing, enabling efficient mobile platforms for imaging, analysis, and interpretation [4]. In this thesis, we explore methods for detection of plankton. The aim is to segment relevant information from irrelevant information in images captured by the AUV. A viable method for in-situ attention segmentation has two requirements. It must be computationally efficient enough for real-time applications, and it must provide high segmentation performance on images captured by the AUV.

1.2 Research questions

We define four research questions for the work in this thesis. The research questions are answered in the discussion section of the thesis.

What is the best performance we can achieve using deep learning models for attention segmentation on our planktonic dataset?

Deep convolutional neural networks have in recent years provided significantly increased performance for most computer vision tasks [5]. This also applies for the computer vision task of image segmentation. Therefore we implement and compare selected deep learning methods to perform attention segmentation on our planktonic dataset. For the method to be viable for real-time applications, it must provide fast run times for predictions. For real-time applications, a compromise of lower performance and faster run times often has to be made. Our first research question focuses solely on segmentation performance. This way we can gain insight on the magnitude of the compromise we must make, with regards to segmentation performance.

What is the best deep learning model for real-time attention segmentation on our planktonic dataset?

This research question focuses on finding the best model for real-time in-situ attention segmentation on the planktonic dataset. We want to find a method that can provide high segmentation performance while at the same time yielding low run times for predictions. We test different networks using different encoders and compare segmentation accuracy and run time for the selected variations.

What is a suitable loss function for our segmentation task?

Neural networks are trained with the aim of minimizing a loss function. In supervised learning, the loss function is calculated during the training process by comparing the predictions from models to true labels of training data. Loss functions differ in the way the error is calculated, and when choosing a loss function we must consider the nature of the task at hand. For instance, for classification tasks with imbalanced datasets, it can be advantageous to use a loss function that gives a larger error when models misclassify objects belonging to rare classes. In this thesis, we test different loss functions to find one that is suitable for the segmentation problem.

Can segmentation results improve by generating additional labelling for training data using Mixture Gaussian based segmentation (MOG)?

Labelling our data is demanding, and must be carried out by a domain expert, a biologist. We have an abundance of available unlabeled images, but only 312 labeled images. Mixture-Gaussian-based segmentation (MOG) [6] [7] can subtract background from foreground without the use of labeled data in images presented as time series. Images in our dataset are captured by the AUV described in [4] several times every second, and can therefore be regarded as images presented as time series.

Deep neural networks require large labeled datasets in training. To overcome this, data augmentation is often used to generate additional training data from data in the training set. In this thesis, we use MOG to generate additional labelling for training data, which can be viewed as a special case of data augmentation. We train networks on a mixture of images labeled by a domain expert and images labeled with MOG. We compare segmentation results of networks trained on only manually labeled images with results from networks trained on a mixture of manually labelled images and images labelled using MOG. This comparison is carried out to determine if adding MOG labeled images to our training dataset can yield improved segmentation.

1.3 Motivation

To detect and localize plankton in images captured by the AUV described in [4], we perform attention segmentation. In [4] the goal is to perform real-time in-situ detection and classification of plankton. One approach towards detection and classification of plankton is instance segmentation. In his master thesis "Object detection and instance segmentation of planktonic organisms using Mask R CNN for real-time in situ mage processing" [8], Sondre Bergum explored the use of Mask R-CNN for instance segmentation in planktonic images. The network was trained to detect and classify two different types of plankton. However, with instance segmentation, only the predefined classes of plankton are detected.

There are two reasons for implementing attention segmentation methods for the planktonic image dataset. The first reason is that by performing attention segmentation we can detect all species of plankton, not only the plankton belonging to predefined classes. The second reason is that the output masks from attention segmentation might improve classification accuracy if used as a backbone in plankton classification. A viable method for real time attention segmentation must provide fast run times and accurate predictions of segmentation maps.

1.4 Contribution

The main contribution of this thesis consists of comprehensive testing and evaluation of deep learning network architectures for attention segmentation. We implement and test five deep learning networks for attention segmentation, and we test several loss functions to find a suitable loss function for the segmentation task. We then test five different pre-trained encoders as backbones for network architectures. From our experiments, we recommend two models for attention segmentation on the planktonic dataset. The first model achieves the best dice score in the conducted experiments, and the second model provides low prediction run-times and high dice score. In addition to dice score, precision and recall are used as evaluation metrics to measure model performance. Furthermore, prediction run-time, which is the time it takes for a model to create a segmentation map output from an input image, is measured.

The second contribution is to create additional labelling for training data using Mixture-Gaussian-based segmentation. Images presented as time series are used as input to the Mixture-Gaussian-based segmentation, and dilation and erosion is used to remove noise in the outputs. We train deep learning networks on the original dataset containing only manually labeled images, and on an enlarged dataset containing both manually labelled images and images labelled through Mixture-Gaussian-based segmentation. We finally compare results for the models trained on the different datasets.

1.5 Outline

This thesis is structured as follows. In chapter 2, background theory relevant for deep learning methods for attention segmentation is covered. In chapter 3, methods used in this thesis are detailed. Chapter 4 covers datasets for which attention segmentation is carried out. Chapter five discusses the proposed framework and implementation details for experiments. Chapter 6 details the experiments carried out in this thesis. Chapter 7 provides results from experiments, and chapter 8 discusses the results. Finally chapter 9 concludes the thesis, and chapter 10 highlights possible future research directions.

Chapter 2

Theoretical background

In this chapter, we discuss relevant background theory for image segmentation using deep learning methods. We first discuss the problem of image segmentation. Since deep learning is a sub-field of machine learning, we define the term "machine learning" in section 2.2. Next, we go in detail on deep learning and focus specifically on theory relevant for computer vision tasks using deep learning methods.

2.1 Image segmentation and attention segmentation

Image segmentation is the process of dividing an image into multiple segments or objects [9]. This can be very useful for object recognition tasks because in object recognition tasks processing an entire image is often not efficient [10]. This especially applies in images where objects make up a small proportion of the image, such as in the planktonic dataset for which we perform attention segmentation in this thesis. By the term attention segmentation, we refer to pixel by pixel segmentation of foreground from background.

In recent years deep learning methods for image segmentation have been very successful. However, there also exist many non-deep learning segmentation methods and applications. This includes a broad variety of segmentation methods based on edge detection, morphological operators, thresholding, and color space extraction [11]. Segmentation based on edge detection utilize discontinuity in local features to extract maps with edges of objects [11]. Segmentation based on color space maps different colors into different classes. The segmentation algorithms based on thresholding techniques, measure the pixel intensity on the grayscale before classifying the scales based on threshold values. [11].

Due to the success of deep learning models in many computer vision applications, there has been a large amount of work in recent years aimed at developing deep learning models for image segmentation[12]. We list and briefly explain four important image segmentation tasks aimed at labelling each pixel of an image to a certain class. These segmentation tasks are often solved using deep learning models. In figure 2.1 the segmentation tasks are illustrated.

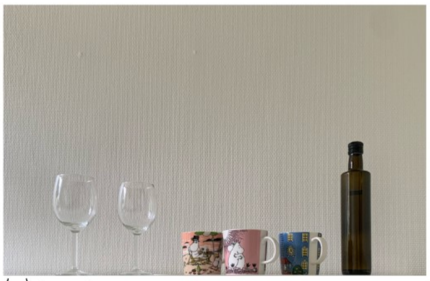
- **Semantic segmentation**
  Semantic segmentation assigns all pixels to a class, but does not differentiate between separate objects of the same class.
- **Instance segmentation**
  Instance segmentation places pixels into a class, but unlike semantic segmentation distinguishes between separate objects of the same class. Generally, instance segmentation ignores pixels not belonging to countable objects.
- **Panoptic segmentation**
  Panoptic segmentation is a mixture of semantic segmentation and instance segmentation. Panoptic segmentation places both a categorical label on pixels and a label indicating which instance of that class the pixels belong to.
- **Attention segmentation**
  Attention segmentation is in this thesis defined as an image segmentation technique that places all pixels of an image in one of two classes, foreground or background. This approach aims at emphasizing relevant information while all irrelevant information is considered as background.

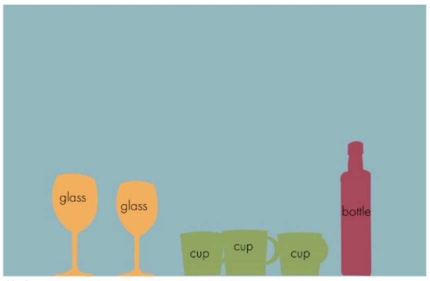
2.2 Machine Learning

Essentially, a machine learning algorithm is an algorithm that improves at solving a certain task through experience [13]. Many of the tasks solved by machine learning algorithms fall under one of two categories; classification tasks, or regression tasks. For classification tasks, the goal is to place the input in the correct category out of *n* predefined categories. For instance, given an image of a cat, place the correct label, "*cat*", on the input image. For regression problems, the algorithm predicts a numerical value given some input. For instance, given the temperature and time of day in a city, predict the electric energy consumption for the city for the next hour. Other categories of tasks that can be solved by machine learning algorithms include transcription, machine translation, imputation of missing values, denoising, clustering, and ranking [13] [14].

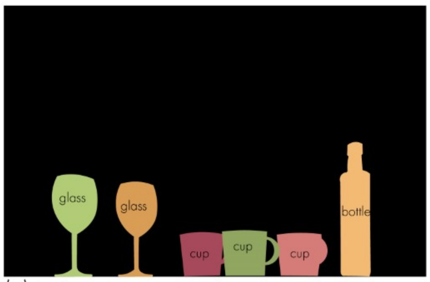
Machine learning algorithms can be divided into categories based on whether or not labeled training data is available, and methods of learning patterns in data.

- Supervised learning uses a set of labeled examples in training, and make predictions for all unseen points [14].
- Unsupervised learning uses exclusively unlabeled data in training. The goal is to find useful patterns in the unlabeled data, for instance by grouping similar instances together in a process called clustering [14]
- Reinforcement learning methods learn through interactions with the environment. Based on these interactions the learner receives awards, with the

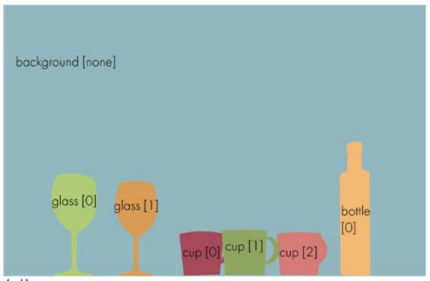

(a) input image



(b) semantic segmentation



(c) instance segmentation



(d) panoptic segmentation

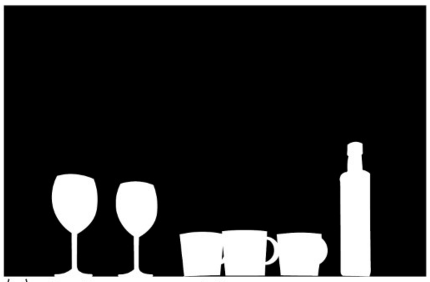

(e) attention segmentation

Figure 2.1: Segmentation techniques

aim of maximizing the total rewards. [14]
- Semi-supervised learning uses a mixture of labeled and unlabeled data in training, typically in situations where there is an abundance of unlabeled data, but labels are expensive to obtain. [14]

Other learning scenarios than the above mentioned could also be encountered [14].

2.3 Deep learning

Deep learning exploits several layers of non-linear information for pattern analysis, classification and feature extraction and transformation [15]. Deep learning is a sub-field within machine learning, that generally uses artificial neural network architectures. The layers correspond to levels of concepts, where higher level concepts are defined from lower layer concepts [15]. Deep learning artificial neural networks are capable of solving many complex multivariate and non-linear modelling problems [16].

2.3.1 General artificial neural network architecture

Artificial neural networks consist of neurons and connections between the neurons. Each connection has a weight associated with it, and each neuron has bias associated with it. These weights and biases are updated during training [17]. The first layer is called the input layer, and the final layer is called the output layer. Between the input and output layers are hidden layers. The depth of a network is defined by the number of layers, and the width of a layer is defined by the number of neurons in the layer [13]. The term "*deep neural networks*" refers to neural networks with several hidden layers. A general neural network architecture is illustrated in figure 2.2.

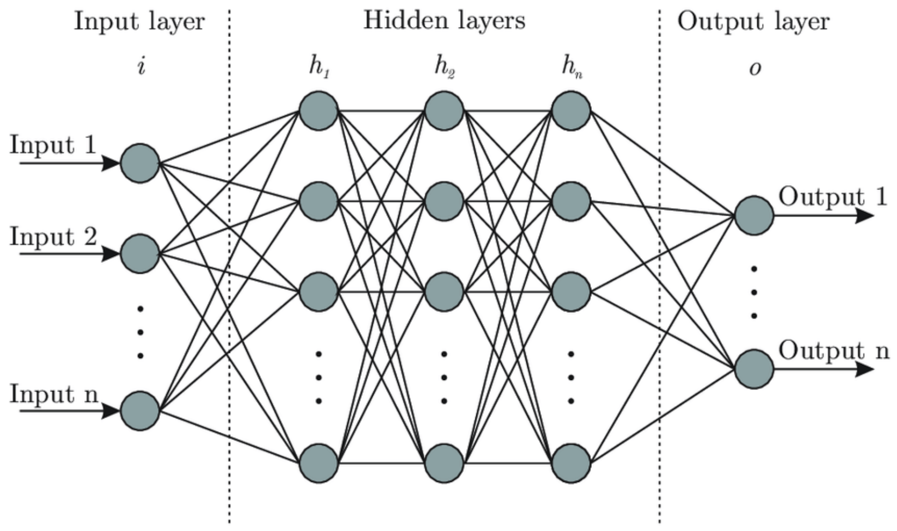


Figure 2.2: General artificial neural network architecture, as illustrated in [16]

2.3.2 Feedforward neural networks

Feedforward neural networks define a mapping $\mathbf{y} = f(\mathbf{x}, \theta)$, where θ represents learnable parameters, $\mathbf{x}$ represents the input and $\mathbf{y}$ represents the output [13]. As opposed to recurrent neural networks, feed forward neural networks do not have feedback connections, thus they can be represented as directed acyclic graphs. A feed forward network with 4 layers can be expressed as $f(\mathbf{x}) = f^{(4)}(f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}))))$, where $f^{(1)}$ represents the first layer, $f^{(2)}$ the second layer, and so on [13]. In a fully connected feed forward neural network the activation of neurons in the hidden layers can be described in a vectorized form as in equation 2.1 [17]. Fully connected, in this context, means that every neuron is connected to all neurons in the previous layer.

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{2.1}$$

As defined in [17], a^l is the activations in the l^{th} layer, a^{l-1} is the activations in the $(l-1)^{th}$ layer, and w^l represents the weights on the connections between the layers. b^l represents the biases associated with the neurons in the l^{th} layer, and σ is an activation function.

2.3.3 Activation functions

Activation functions add non-linearity to neural networks. This enables networks to learn complex patterns in data. Without activation functions the output from neural networks would be simple linear functions [18]. Activation functions transform an input signal to an output signal that can be passed to the next layer in the network.

The sigmoid activation function is commonly used in the output layer of neural networks. It places all values between 0 and 1 and can be expressed in the following manner [19].

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Another commonly used activation function, the hyperbolic tangent activation function (Tanh), is zero centered and outputs values between -1 and 1. It can be expressed in the following manner [19].

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{2.3}$$

The Tanh function is preferred over the sigmoid for use in hidden layers, as it gives better training performance [19]. However the use of both of these activation functions in deep networks can cause the vanishing gradient problem to occur in recurrent networks, and feed-forward networks with many layers [19] [20]. The gradients of loss functions can become very small, as more layers are added in a network. As a result, network weights do not get updated during training, and the model fails to learn. This is called the vanishing gradient problem [21]. To combat the vanishing gradient problem, the rectified linear unit (ReLU) activation function is often used. This activation function outputs 0 for all negative inputs. For positive inputs, it is a simple linear function. ReLU is the most widely used activation function in state-of-the-art deep neural networks [19], as it offers faster learning and better generalization than the sigmoid and the Tanh activation functions. The ReLU activation function can be expressed in the following manner [18].

$$f(x) = \max(0, x) \tag{2.4}$$

In addition to the above mentioned, several other activation functions are available. Some examples are softmax, softsign, softplus, ELiSH and swish [19]. In addition, there are some variations of many of the mentioned activation functions. For instance the leaky ReLU, which adds a small negative slope to the ReLU [19].

2.3.4 Loss functions

Supervised learning of neural networks is a process of updating weights to reduce the error in training data [22]. In order to calculate the error of training data, loss functions[1] are used. Loss functions consider the values of the labels of training data and the predictions made by a model during training to calculate an error. When choosing a loss function, we must consider the nature of the task at hand [23]. For instance, if training a network to detect cancer in patients, we want to avoid false negatives. Therefore, for that task, a loss function should give a larger error when detecting false negatives than when detecting false positives. This way, the network learns to prioritize avoiding false negatives, and the chance of predicting false negatives can be reduced for unseen data.

A common loss function used in regression problems is the Mean Squared Error (MSE). This loss function simply calculates the square of the difference between the predicted output, and the true labels of the training data. The MSE loss function can be expressed as in equation 2.5 [22], where Y_i is the label, $\hat{Y}_i$ is the predicted output.

$$MSE = \frac{1}{2n} \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i \right)^2 \tag{2.5}$$

The binary cross entropy (BCE) loss function is commonly used for classification problems. It is a special case of cross entropy and gives the average loss of entropy between the predicted- and true labels [24]. The labels are usually either 1 or 0, indicating if an instance belongs to a class or not, and predictions are numbers between 0 and 1, indicating the confidence the model has that an instance belongs

[1]Loss functions are also known as cost functions or objective functions.

to the class. In equation 2.6, a mathematical expression for BCE is shown. Here y_i is the label, $\hat{y}_i$ is the predicted output.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{2.6}$$

Many loss functions exist in the literature. Some examples of commonly used loss functions are Hinge loss [25], Huber loss [26], and KL-divergence [27]. Some other loss functions are designed to help solve more specific tasks in machine learning. In chapter 3, we further discuss loss functions in the context of image segmentation.

2.3.5 Backpropagation and Optimizers

The process for updating weights and biases in a neural network during training to minimize the loss function is called backpropagation. Errors calculated at the loss function are propagated throughout the network to update parameters during training [13]. Optimization strategies are used to effectively minimize the loss function. Most optimizers used in deep learning are built on gradient descent, which calculates the gradient of the average loss of all samples in the training set [13]. In practice this often requires large memory, and therefore mini-batch gradient descent is often used instead. Mini-batch gradient descent computes the gradient of the average loss for sub-samples (batch) of the training data at each weight update to reduce memory requirements. The number of samples to include in each batch is determined by the batch size, which is a hyper-parameter. The learning rate is another hyper-parameter that determines the magnitude of the updates by scaling the gradient. Another optimizer, the Adam optimizer [28], employs efficient methods for updating the learning rate during training, and to accelerate convergence in relevant directions.

2.3.6 Generalization over-fitting, and under-fitting

In machine learning tasks, the ability of models to generalize well is very important. Models should learn the general patterns in the training data to be able to make good predictions for validation and test data [29]. Under-fitting refers to the situation where models perform poorly on both the training data and unseen data. In deep learning, it is common to train models over several iterations. These iterations are called epochs and one epoch refers to one iteration of training over the entire training set. By training over several epochs, networks can better learn patterns in the training data to help avoid under-fitting. Over-fitting, on the other hand, refers to the situation where models are well adapted to the training data, but performance on unseen data is poor. A common cause of over-fitting is over-training meaning the model is trained over too many epochs [30]. By monitoring the performance on the validation set at every epoch during training, we can stop training when performance on the validation set decreases. Alternatively, we can save a model at every epoch, and select the model that has the best performance on the validation set after training over all epochs is completed. Some other techniques to avoid over-fitting include dropout [31], batch-normalization [32] and One-shot and Zero-shot learning [33] [34].

A method to avoid over-fitting that we use in this thesis is Transfer learning [35]. In transfer learning, a network is trained on a big dataset, such as ImageNet [36]. The resulting weights can then be used as initialization weights in a new task [30]. For computer vision tasks this is often very useful, as different datasets often share many of the same low-level characteristics, which are best learned with big datasets [30].

2.3.7 Data augmentation in computer vision

Data augmentation increases the size of training datasets by generating additional training data. The original training data is typically used as the basis for generating additional data. Data augmentation is commonly used in computer vision to improve generalization for deep learning models. This especially applies when there are few labeled training images available [30]. A model with little available training data tends to over-fit. Data augmentation increases the number of data points for training, decreasing the distance between the training set and the validation set. This often yields improved model performance on the validation set [30].

Data augmentation can be carried out through data warping or oversampling. Data warping changes the characteristics of an image through techniques like geometric transformations, random erasing, neural style transfers, and color transformations [30]. Oversampling involves creating synthetic data exploiting characteristics in the original training data. Methods for oversampling include mixing images, feature space augmentation, and generating data using generative adversarial networks (GANs) [30].

Data augmentation using basic image manipulations

Two widely used groups of data warping manipulations are geometric transformations and color space transformations. Geometric transformations change geometric properties of the training images to make the models more robust to changes in position and orientation [37]. Some examples of geometric transformations are rotation and flipping. Color space transformations involve changing the color space within training images to make models more robust towards variations in lighting and color [37]. Oversampling techniques using basic image manipulations include random erasing and mixing images. Random erasing is done by randomly selecting patches of images assigning all pixels values within the selected patches randomly or with predefined values. Random erasing is done to overcome overfitting due to some objects or parts of images being unclear [30]. Mixing images combines sections of images into synthetic images [38]. Patches of images are extracted and pasted together to generate new images.

Data augmentation using deep learning techniques

Neural networks can map high dimensional inputs into lower dimensional representations [30]. Lower dimensional feature maps can be isolated and extracted. This opens up possibilities for using neural networks for data augmentation. Generative adversarial networks (GANs) can be used to generate artificial instances from a dataset while retaining similar characteristics of the original dataset [30]. Another method, called neural style transfers can recreate an image so that it is displayed in a different style, while still retaining the original image motive [30].

2.3.8 Convolutional neural networks

Convolutional neural networks (CNNs) are inspired by the visual perception mechanism of humans and animals [39]. Through convolution operations, a specialized kind of linear operation, CNNs can extract feature representations from an input [39] [13]. CNNs can be defined as neural networks that use convolution in place of general matrix multiplication in at least one of their layers [13]. In recent years, CNNs have conquered most computer vision fields, for their ability to extract useful feature representations in real world applications with complicated images [5].

In the convolutional operation, a kernel, which is a small array of numbers, is applied across the input. The input is in the form of an array, called an input tensor [40]. By calculating an element-wise product between every element of the kernel and the input tensor, a feature map is generated. The feature map represents the output of the convolutional operation at every position in the input tensor [40]. Convolutional operations enable extraction of contextual information in images. The convolutional operation is illustrated in figure 2.3.

The outputs from convolutional operations are usually passed through an activation function. Following the activation function is usually a pooling layer. Pooling layers provide down-sampling operations, reducing the dimensions of feature maps [40]. The most common pooling operation is max pooling, where the output is the maximum value within patches extracted from input feature maps [40].

The fully convolutional networks (FCN) for semantic segmentation, proposed by *Long et al.* (2015) [41], showed great potential for the use of convolutional neural networks for image segmentation. FCN uses an encoder-decoder structure, where the encoder extracts features, and the decoder creates a segmentation map output of the same dimensions as the original input image. Skip connections from

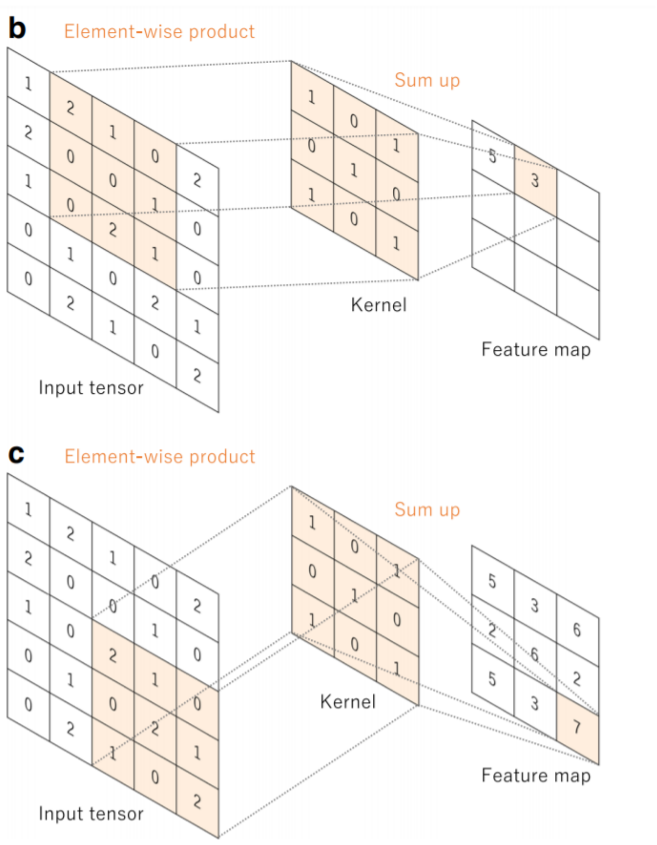


Figure 2.3: The convolutional operation as illustrated in [40]

the encoder to the decoder aid the decoder in creating the segmentation map output. The encoder consists of convolutional layers and pooling layers, while the decoder consists of convolutional layers and up-sampling operations. The Up-sampling operations increase the size of feature maps. Through skip connections, feature maps are passed from layers in the encoder to provide spatial information to the decoder, thus allowing the decoder to produce finer segmentation. The work presented in [41] popularized the use of convolutional neural networks for segmentation tasks. The neural networks used for attention segmentation in this thesis all build on underlying ideas presented in [41].

Chapter 3

Methods

In this chapter, we go in detail on the deep neural networks used for attention segmentation in this thesis. We then discuss encoders, which are used as backbones for networks for attention segmentation. Next, we discuss loss functions and evaluation metrics used in the experiments. Finally, we discuss Mixture-Gaussian-based segmentation which is used in this thesis to generate additional training data.

3.1 Deep learning models for attention segmentation

We conduct experiments on- and implement five different neural network architectures for attention segmentation. The network architectures used in this thesis all build on the fully convolutional network (FCN) architecture proposed in [41]. The networks all use an encoder-decoder structure, where the encoder extracts spatial information in images, and the decoder creates the segmentation map output.

3.1.1 U-net

U-net has its name from the u-like structure of the network architecture. It consists of an encoder (the contracting path), a decoder (the expanding path) and skip connections between these. The encoder consists of repeated 3x3 convolutions, where each convolution is followed by a 2x2 max pooling layer and an activation function, a rectified linear unit (ReLU), [42]. For every convolutional layer in the encoder, the number of feature channels is doubled. In the decoder, an up-sampling operation is followed by two 3x3 convolutions at every step. Each of the convolutions are followed by a ReLU. For every up-sampling operation, the number of feature channels is halved [42]. Skip connections between the encoder and decoder ensure fine grained details can be recovered in predictions, as pattern information extracted in the encoder is passed to the decoder [9]. The FCN, which the U-net builds upon does not consider global contextual information in an efficient way [9], but U-net is more efficient than FCN at capturing and exploiting contextual information. U-net was by the authors in [42] used for a biomedical image segmentation task with little available training data. In [42] data augmentation is used to generate additional training data, and it is shown that U-net obtains impressive segmentation results for the biomedical segmentation tasks. The network architecture of U-net is shown in figure 3.1.

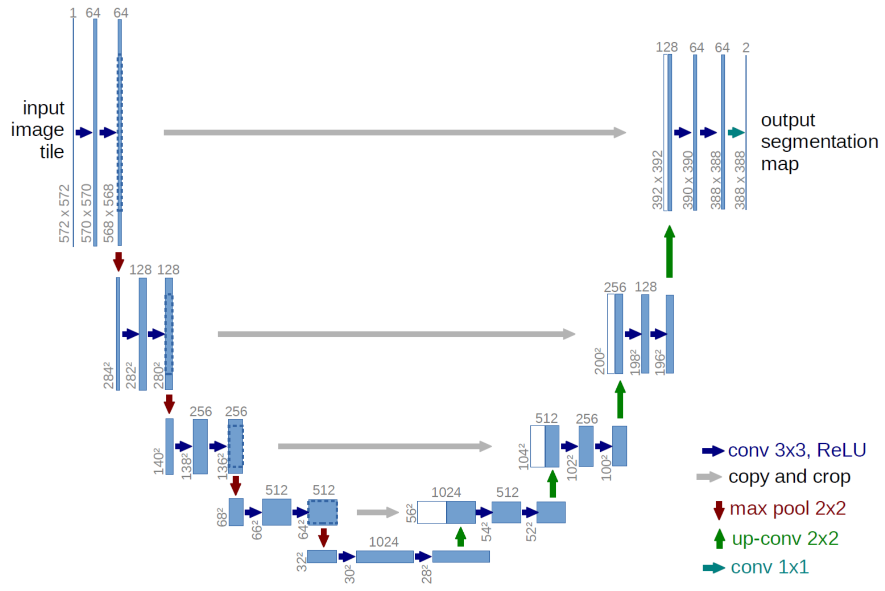


Figure 3.1: U-net architecture as illustrated in [42]

3.1.2 U-net++

The U-net++ [43] is similar to the U-net [42] in that both networks has the same symmetric encoder and decoder structure. However, in the U-net++, the encoder and decoder are connected through a series of nested dense convolutional blocks [43]. For the dense nested convolutional blocks, each layer receives inputs from all preceding layers and passes feature maps to all subsequent layers [44]. The underlying idea for the U-net++ is to make encoder and decoder feature maps more similar, under the assumption that this would lead to improved segmentation. The u-net++ is in [43] trained on four different biomedical image datasets, yielding improved segmentation over U-net [42] for all datasets. In [43], the U-net++ is tested with and without deep supervision, and the deep supervision is implemented as proposed in [45]. The U-net++ architecture is shown in figure 3.4.

3.1.3 Deeplab V3+

Deeplab V3+ uses dilated convolution, also known as atrous convolution, to increase the size of receptive fields without increase in computational cost [9]. The

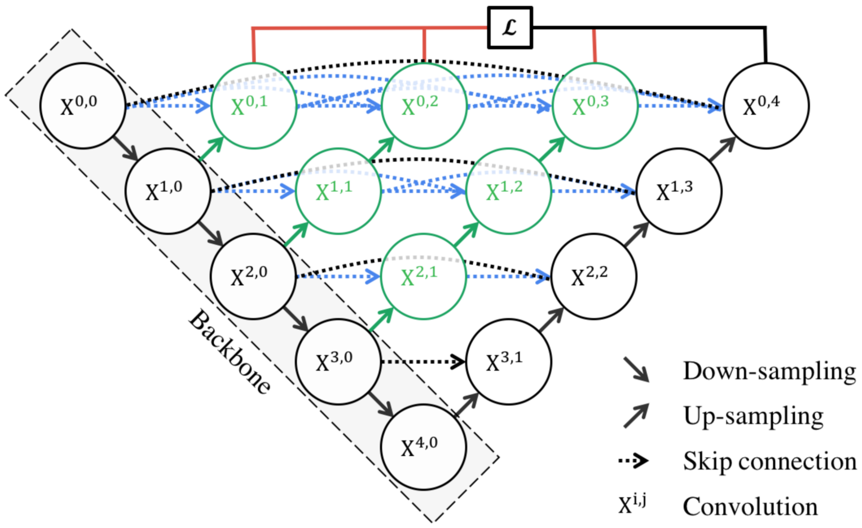
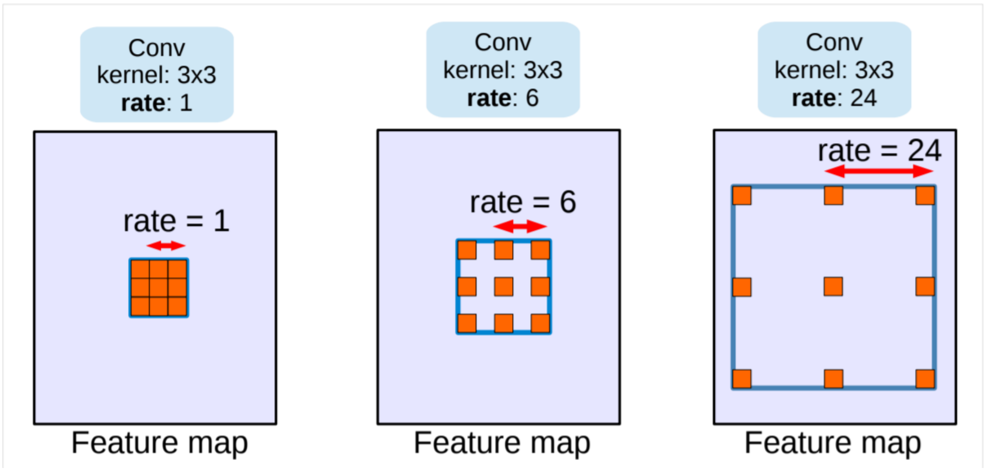


Figure 3.2: U-net++ architecture as illustrated in [43]

dilated convolution is illustrated in figure 3.3. The rate controls the size of the receptive fields in dilated convolutions. In [46], dilated convolution is done through atrous spatial pyramid pooling [47], where several dilated convolutions using different rates are carried out in parallel, and the resulting feature maps of the dilated convolutions are fused at the output. This way convolutional features at different scales can be extracted [46]. Deeplab V3+ [46] is similar to Deeplab V3 [48]. However Deeplab V3+ [46] extends Deeplab V3 [48] by adding a simple and effective decoder module to refine segmentation results especially along object boundaries [46]. Deeplab V3+ is one of the best networks for semantic segmentation on the Cityscapes dataset [49].



Conv kernel: 3x3 **rate**: 1
rate = 1
Feature map
Conv kernel: 3x3 **rate**: 6
rate = 6
Feature map
Conv kernel: 3x3 **rate**: 24
rate = 24
Feature map

Figure 3.3: Dilated convolutions as illustrated in [48]

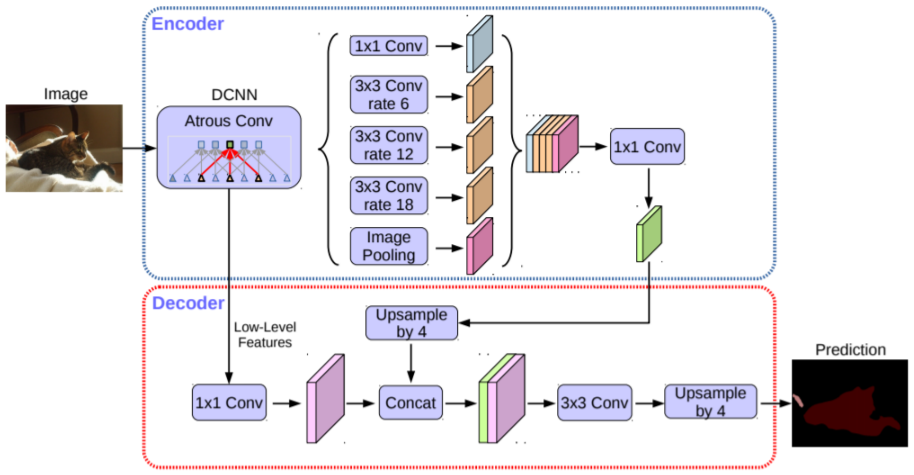


Figure 3.4: Deeplab V3+ architecture as illustrated [46]

3.1.4 Pyramid attention network (PAN)

Pyramid attention network (PAN) [50] proposes two modules embedded in an encoder-decoder network for segmentation. Dilated convolutions, as used in the Deeplab V3+ [46] may cause local information important for consistency in feature maps being overlooked [50]. The Feature Pyramid Attention (FPA) module fuses features of three different scales, to extract features at different scales. FPA applies 3x3, 5x5, 7x7, in a pyramid structure with downsampling and upsampling operations. The FPA module is shown in figure 3.5. The Global Attention Upsample (GAU) module proposed in [50] provides global context in the decoder as guidance for low level features, through the use of Global average pooling. The GAU module is shown in figure 3.6. In the Pyramid attention network architecture, the FPA module is a center block between the encoder and the decoder structure. The GAU module is used in the decoder structure of the network. The full network architecture is shown in figure 3.7.

3.1.5 LinkNet

LinkNet [51] is specifically designed for computationally efficient segmentation, to make the network more suitable for real-time segmentation tasks. LinkNet has an encoder-decoder structure and passes information straight from the encoder to the decoder. This way, the network can preserve information that would be otherwise lost at each level of the encoder, while no additional parameters and operations are needed for relearning this information [51]. By efficiently minimizing the number of network parameters, LinkNet provides fast and accurate segmentation. The network architecture is shown in figure 3.8.

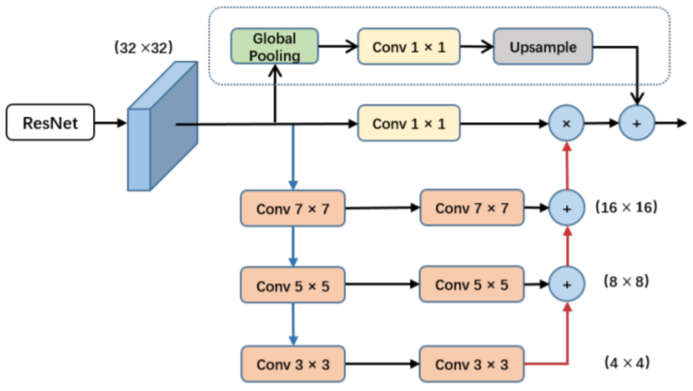


Figure 3.5: Feature Pyramid Attention module as illustrated in [50]

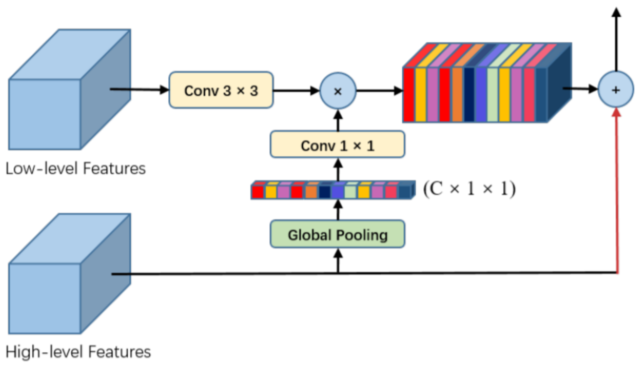


Figure 3.6: Global Attention Upsample module as illustrated in [50]

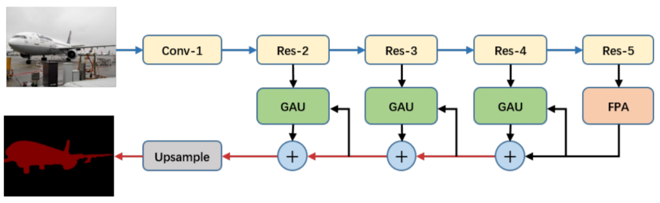


Figure 3.7: Pyramid attention network architecture as illustrated in [50]

3.2 Pretrained encoders

Many of the same low level characteristics are shared by different image datasets. For a model to learn low level characteristics, large datasets are often needed [30]. By training encoders on large datasets, such as ImageNet [36] general low level characteristics can be learned. Pre-trained encoders can be applied as backbones for various computer vision tasks. This is very useful for tasks with little available

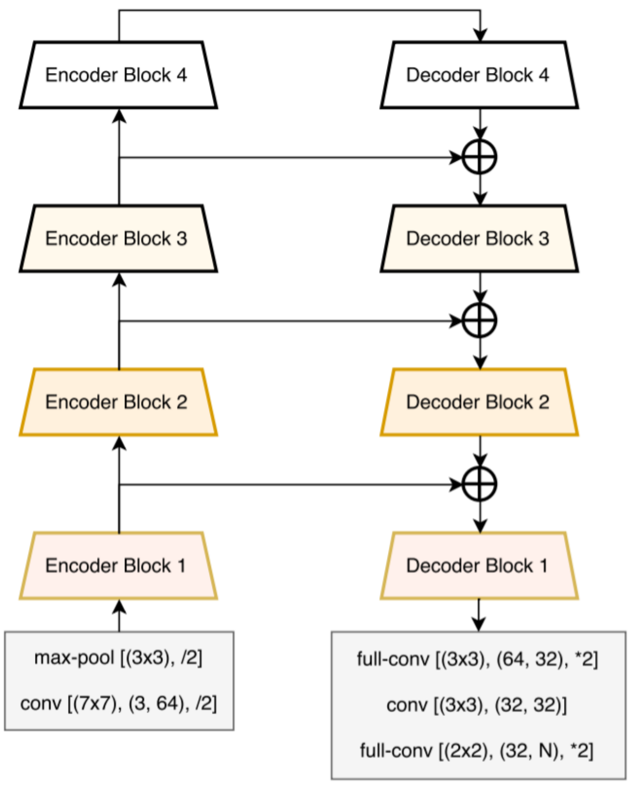


Figure 3.8: LinkNet architecture as illustrated in [51]

training data. In this thesis, we perform attention segmentation on a small dataset and apply encoders pre-trained on ImageNet [36] in all our experiments.

Encoders differ in computational efficiency and number of parameters. One of the research questions in this thesis focuses on finding the model that produces the best segmentation maps for our dataset. We do experiments using selected pre-trained encoders, to find which encoder-network pairing produces the best segmentation maps. Another research question in this thesis focuses on finding an efficient method for real-time attention segmentation. We do experiments using several light pre-trained encoders, to see if we can reduce prediction run times without significantly reducing segmentation performance. In this section, we discuss encoders used in our experiments. These are ResNet, EfficientNet, and MobileNet v2.

3.2.1 Deep residual networks (ResNet)

In [52] it is proposed Deep Residual networks for image recognition. Very deep neural networks can be difficult to train, and it is claimed in [52], that the use of a residual learning framework eases training of deep neural networks. Residual networks use shortcut connections between layers, which increases the depth of the networks, without additional parameters or complexity [53]. Deep residual networks have been extensively used in the literature, with great success for a variety of deep learning applications [54]. In segmentation tasks, ResNet is often used in the encoding of images to extract dense features. The authors in [50] achieve the best segmentation performance for their model using a deep residual network with 101 layers (ResNet-101) pre-trained on ImageNet [36] in the encoder part of their network. In figure 3.9, a building block of ResNet is shown. In the literature, variations of ResNet are named by the number of convolutional layers. For instance, ResNet-101 has 101 convolutional layers, and ResNet-34 has 34 convolutional layers.

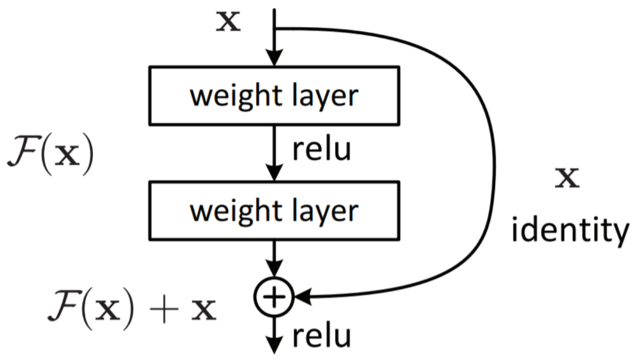

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.9: A building block for residual learning as illustrated in [52]

3.2.2 EfficientNet

For many applications of convolutional neural networks, the availability of computational resources often limits model complexity. EfficientNet [55] offers effective scaling of model complexity so that models can be better adapted to limitations in computational resources. In addition, Efficient-net provides efficient model scaling for situations where more resources become available. The complexity of ResNet [52], can be scaled up or down by changing the depth. EfficientNet offers a more flexible scaling by scaling in three dimensions. These scaling dimensions are depth scaling, width scaling, and resolution scaling. Deeper networks are often able to capture richer and more complex features, and wider networks are often easier to train and can capture more fine grained features [55]. Resolution scaling refers to adjusting the resolution of input images. Using higher dimensional input images tends to lead to more fine grained patterns being captured [55]. The different scaling dimensions are dependent on each other, and in [55] a "compound coefficient" is proposed to effectively scale networks at different dimensions. Eight different architectures with different complexities are proposed in [55] ranging from EfficientNet-B0 to EfficientNet-B7, where B0 has the lowest complexity and B7 has the highest complexity. Network scaling dimensions in

EfficientNet are illustrated in figure 3.10.

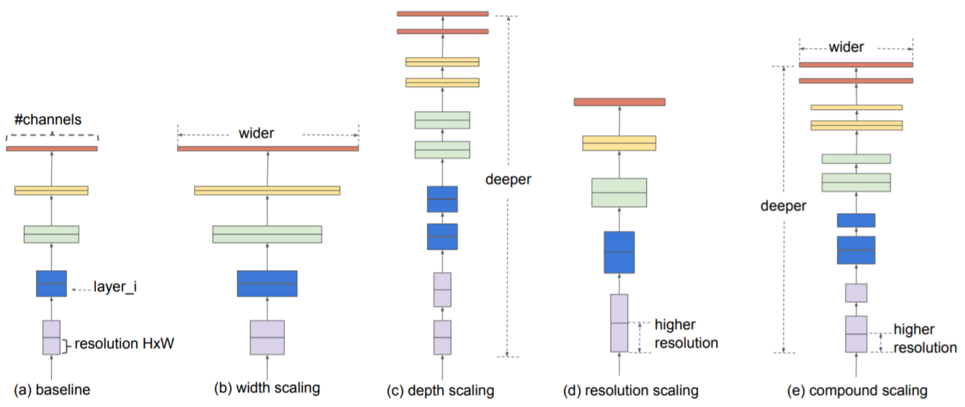


Figure 3.10: Scaling dimensions in EfficientNet as illustrated in [55]. Compound scaling combines all scaling dimensions

3.2.3 MobileNet v2

MobileNet v2 [56]is designed for mobile platforms, to have low computational complexity, yet provide high performance. It uses depth-wise separable convolutions to split convolutions into two layers replacing the standard convolutional layer [56]. Depth-wise separable convolutions apply a depth-wise convolution before a point-wise convolution. The depth-wise convolutions apply convolutions for a single input channel at the time, as opposed to full convolutional operators that apply convolutions to all channels. The point wise convolution follows the depth-wise convolutions and computes linear combinations of the input channels to build new features [56]. Together these layers make up Depth-wise separable convolutions, which can replace standard convolutional layers at a lower computational cost. In the MobileNet v2 architecture [56] the depth-wise convolutions use linear bottleneck shortcut connections to form inverted residual blocks.

In low dimensions, the use of non-linear activation functions can cause information to be lost. MobileNet v2 uses linear bottlenecks to expand channel dimensions in low dimensions, to ensure less information is lost. Together, the linear bottlenecks and the depth-wise separable convolution form inverted residual blocks. Layers with a small number of channels are connected, and in these layers no non-linear activation function is used. The inverted residual blocks provide reduce computational complexity while yielding efficient feature extraction. In figure 3.11 a residual block and an inverted residual block is illustrated.

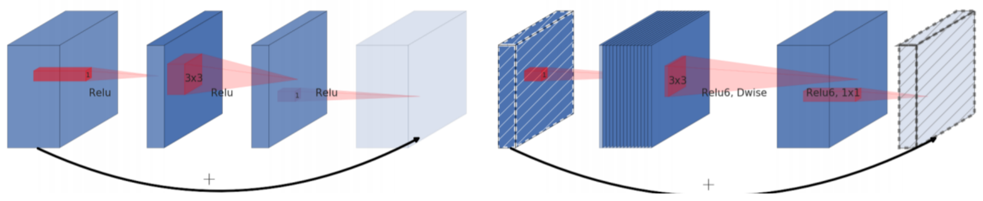


Figure 3.11: Residual block to the left, and inverted residual block to the right. The thickness of the blocks indicate the number of channels. The diagonally hatched layers do not use non-linear activation functions. As illustrated in [56]

3.3 Loss functions

When choosing a loss function for a segmentation task, we must consider the nature of the task at hand. In this thesis, we perform attention segmentation for an imbalanced dataset. The dataset contains many more foreground pixels than background pixels. We wish to obtain a model that detects all foreground particles, and therefore we choose loss functions that prioritize foreground pixels and hard-to-segment samples. We conduct experiments using weighted binary cross entropy loss functions (weighted BCE) [57] and focal loss [58].

Weighted BCE can be modified to prioritize positive or negative samples. The formula for weighted BCE is shown below. When $\beta > 1$ the model prioritizes classifying positive samples correctly reducing the number of false negatives. For $\beta < 1$ the model prioritizes negative samples, reducing the number of false positives [23].

$$\text{Weighted BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i) \tag{3.1}$$

Focal loss prioritizes learning hard examples. p_t denotes the probability of the ground truth class. Focal loss can be formulated as in 3.2. γ decides how much to prioritize hard examples, and when $\gamma = 1$ focal loss works like BCE [23]. α can be changed to make the model prioritize positive or negative samples. In experiments in this thesis, we let $\alpha = 1$ and test focal loss with some variations in the γ parameter.

$$p_t = \begin{cases} \hat{y}, & \text{if } y \geq 1 \\ 1-\hat{y}, & \text{otherwise} \end{cases}$$

$$\text{Focal Loss} = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{3.2}$$

In the equations for weighted BCE (3.1) and focal loss (3.2), y is the true label, and $\hat{y}$ is the model prediction.

3.4 Evaluation metrics

To examine performance of models, we use evaluation metrics. Evaluation of machine learning methods performing classification tasks is not as straightforward as one might think. Accuracy, the most intuitive evaluation metric, simply gives the number of correctly classified instances divided by the total number of instances [59]. For imbalanced datasets, accuracy is not the most efficient evaluation metric. The planktonic dataset used in this thesis is imbalanced, and a model classifying all pixels as background achieves a high accuracy of 98.50, while outputting a useless segmentation map. For attention segmentation tasks, accuracy can be expressed as in 3.3.

$$\text{Accuracy} = \frac{\text{correctly classified pixels}}{\text{total number of pixels}} \tag{3.3}$$

The use of three different evaluation metrics, i.e dice coefficient, precision, and recall, collectively examines model performance effectively [23]. The dice coefficient measures the overlapping between ground truth and predicted output. Precision measures the ability of a model to label only true positives as positives. Precision decreases when the number of false positives increases. Recall measures the ability of a model to label positives correctly but is not affected by the number of false positives. In 3.4, 3.5, and 3.6, we provide the equations for the evaluation metrics. Here TP is true positives, FP is false positives, and FN is false negatives. We refer to foreground particles as positives, and background particles as negatives.

$$\text{Dice Coefficient} = \frac{\text{2TP}}{\text{2TP+FP+FN}} \tag{3.4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{3.5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{3.6}$$

In the literature, the dice coefficient is often used for medical image segmentation, while another evaluation metric, intersection over union (IoU), is commonly used for segmentation tasks in city environments, such as for the Cityscapes dataset [49]. For efficient evaluation, it is sufficient to use either the IoU or the dice coefficient. This is because the dice coefficient has a monotonic increasing relationship to IoU on the interval [0,1]. For all $dice_1 > dice_2$, we will have $IoU_1 > IoU_2$. The IoU is defined in equation 3.7. In equations 3.8, 3.9, and 3.10, we derive the relation between IoU and dice coefficient as in [60].

$$\text{IoU} = \frac{\text{TP}}{\text{TP+FP+FN}} \tag{3.7}$$

We define:

$$\text{TP} = a \qquad\qquad \frac{\text{TP}}{\text{TP+FP+FN}} = \frac{a}{b}$$

Then,

$$\text{IoU} = \frac{\text{TP}}{\text{TP+FP+FN}} = \frac{a}{b} \tag{3.8}$$

$$\text{Dice Coefficient} = \frac{2\text{TP}}{2\text{TP+FP+FN}} = \frac{2a}{a+b} \tag{3.9}$$

Thus,

$$\text{Dice Coefficient} = \frac{\frac{2a}{b}}{\frac{a+b}{b}} = \frac{2\frac{a}{b}}{\frac{a}{b}+1} = \frac{2\text{IoU}}{\text{IoU}+1} \tag{3.10}$$

In this thesis, we choose to use the dice coefficient for evaluation, instead of the IoU, because we regard our planktonic dataset as more related to biomedical semantic tasks, for which the dice coefficient is most commonly used in the literature, than to semantic segmentation in city environments, for which IoU is most commonly used in the literature. In addition, we use precision and recall for performance evaluation. Precision is used to indicate the ability of models to correctly label negatives, and recall is used to evaluate the ability of models to identify positives.

3.5 Mixture-Gaussian-based segmentation (MOG2)

In this thesis, we use Mixture-Gaussian-based foreground-background segmentation (MOG2) to generate additional labelling for our data. MOG2 [6] [7], is based on the original MOG method proposed in [61]. In the MOG method, a mixture of k Gaussian distributions models the individual pixels, where the idea is that each distribution represents different foreground and background colors [62]. The weight on the distributions increases when colors are static at the pixel. This way pixels with low weights are assumed to belong to moving objects and are labeled foreground, while pixels with high weights are assumed to belong to static objects, thus they are labeled background. Values for the parameter k are on the interval [3,5] [62]. MOG2, which we use in this thesis, is an extension of MOG, that uses a variable amount of Gaussian distributions to better capture color complexity in the frames. Since the MOG methods measure movement in frames, they are only applicable to images presented as time series. The images in our planktonic dataset are captured several times each second, and can therefore be regarded as images presented as time series. When using the MOG2 method in our experiments we make sure that the images are presented in the correct order, corresponding to the time images are captured.

3.6 Morphological operations

To remove noise in output images from MOG2, we apply dilation and erosion, two morphological operations [63]. For dilation and erosion, a kernel of dimensions $\mathbf{m} \times \mathbf{n}$ is defined with an anchor point that is usually in the middle of the kernel. For dilation, the pixel value at the anchor point is assigned the maximum value of the pixels within the kernel. For erosion, the pixel value at the anchor point is assigned the minimal value of the pixels within the kernel. In this thesis, we apply dilation followed by erosion, and the operations together in that order is known as closing [63].

Chapter 4

Datasets

In this thesis, we perform attention segmentation planktonic images capture by a Silhouette camera (SilCam), in an Autonomous underwater vehicle (AUV), as described in [4]. The images are captured in the upper water column and labeled by a domain expert, a biologist. The labelling is done using the VGG annotation tool. Labels are stored in a csv file, in the form of x and y coordinates of corners of polygons representing contours of plankton. We convert these coordinates into ground truth segmentation maps [1]. This is done because the deep learning methods used for attention segmentation in this thesis require labels of training data to be in the form of ground truth segmentation maps. In this chapter the datasets used in training of deep learning models for attention segmentation are discussed.

4.1 Original planktonic image dataset

Our dataset consists of 312 images with corresponding ground truth segmentation maps. Both images and segmentation maps have the same 2448×2050 resolution. In figure 4.1, examples of input images paired with their corresponding segmentation maps are shown. Labelling efforts are demanding for our data, and we observe that the labelling does not represent a perfect pixel-wise segmentation map of the input images. The regions labeled as foreground are generally slightly larger than the actual foreground particles. Since the images in our dataset are not perfectly labelled we can not expect any model to output perfect, or near-perfect segmentation maps of foreground and background. We expect output maps of a well performing model to have the same characteristics as the manually labelled segmentation maps, ie. we expect the regions representing foreground particles to be slightly larger than the actual foreground particles. We have also observed that in a small number of images, some planktonic organisms are overlooked by the biologist, meaning they are not labelled at all. We do not expect this to have a noticeable effect on segmentation map outputs from models trained on the dataset,

[1]The code for converting labels to ground truth segmentation maps is available at https://github.com/jonasnb/AILARON-attention-segmentation

as it only applies to very few images and very few planktonic organisms.

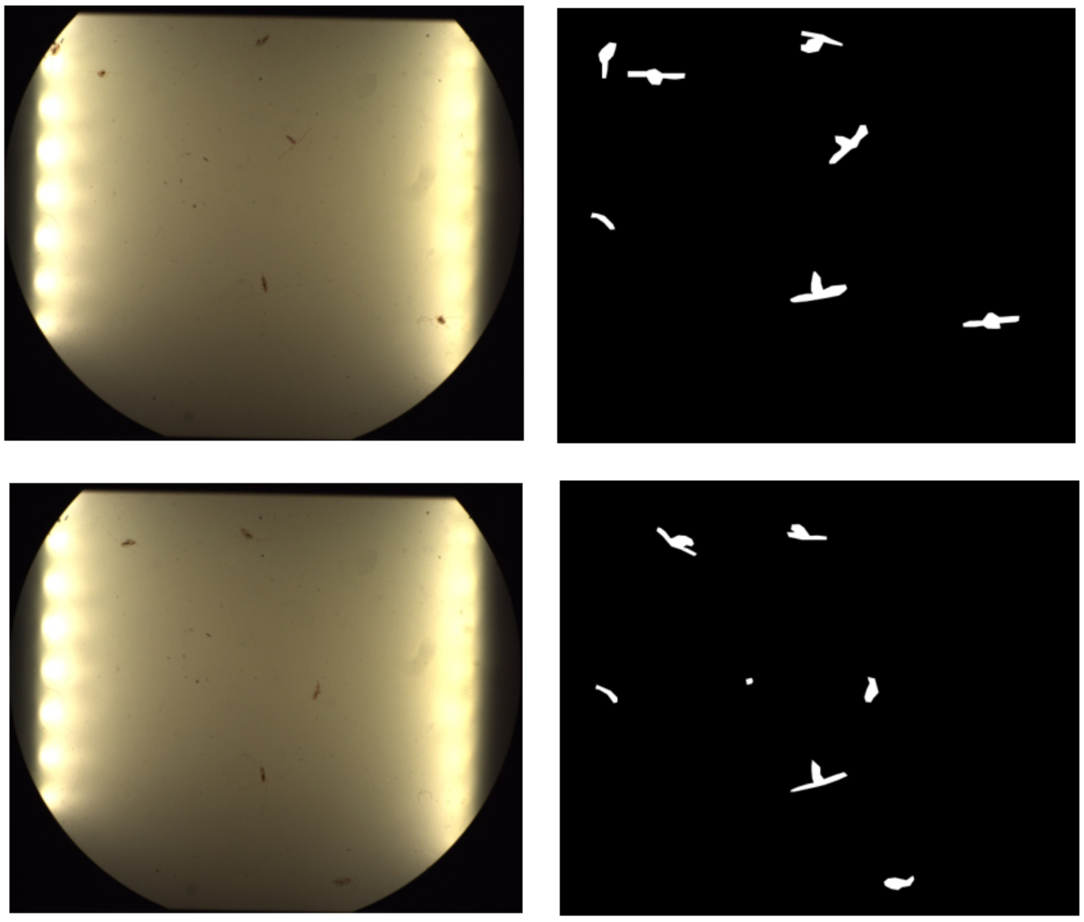


Figure 4.1: Input images and their corresponding manually labeled masks.

4.2 Planktonic image dataset enlarged with MOG2 labeled data

The dataset with manually labeled images is rather small, with only 312 images. We use Mixture-Gaussian-based segmentation (MOG2) to generate additional training data. We use the MOG2 method on all images in our training dataset. We then remove noise in the masks by applying dilation with kernel size 10×10 followed by erosion with kernel size 20×20 [2]. Since this procedure is carried out on the training data, it can be viewed as a special case of data augmentation. In figure 4.2, we show examples of segmentation maps from MOG2

[2]The code for generating additional data using MOG, dilation, and erosion is available at https://github.com/jonasnb/AILARON-attention-segmentation

before and after noise removal. In the noise removal process, the antennas of plankton are lost. We also see that the size of the bodies of the planktonic organisms is smaller after noise removal. Furthermore, we observe that 19% of the masks generated by MOG2 represent very poor foreground-background segmentation, as illustrated with an example in figure 4.3. These masks are not included in our extended training dataset.

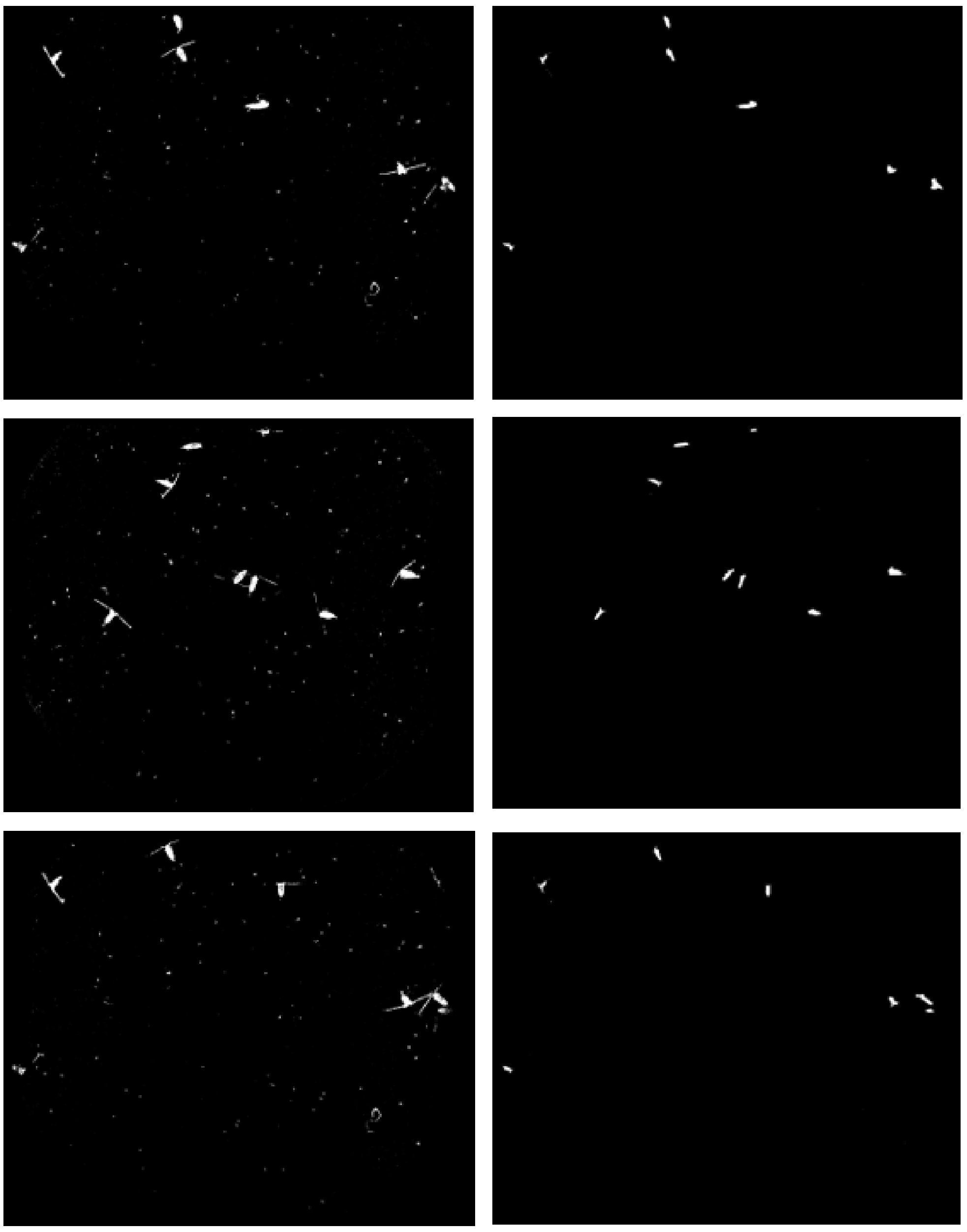


Figure 4.2: Examples of output from MOG2 before and after morphological operations are applied.

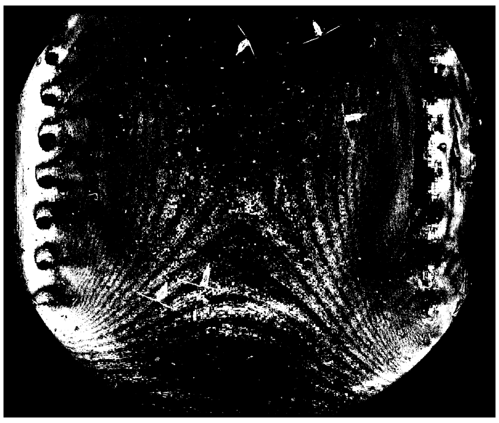


Figure 4.3: Example of bad output from mog2

Chapter 5

Framework and implementation

In this chapter, framework and implementation details for the experiments is discussed. The pipeline for creating segmentation maps is illustrated in figure 5.1. First some initial pre-processing is performed. Then segmentation map predictions are made by a deep learning model, before a post-processing step outputs the final segmentation map prediction.

The source code utilized in this thesis is this thesis is available at `https://github.com/jonasnb/AILARON-attention-segmentation`"[1]

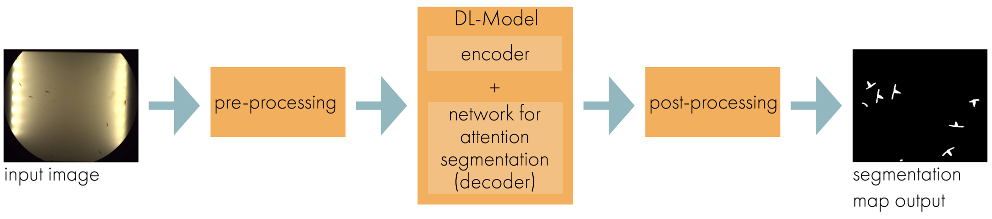


Figure 5.1: Framework for attention segmentation

5.1 Pre-processing

As a pre-processing step, we change the size of all images and masks to 512×512 to make image dimensions compatible with all networks used. This pre-processing is applied for all data before training, and must also be applied when trained models are used to make predictions for unseen data.

[1]Our code for loading data and training networks builds upon the code available at `https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/image_segmentation/semantic_segmentation_unet`

5.2 Deep learning model (DL-Model)

Segmentation map predictions are made from a deep learning model with an encoder decoder structure. The network for attention segmentation acts a decoding scheme for creating segmentation map outputs from features extracted by the encoder. The models are flexible in that it is possible to change either the encoder or the network for attention segmentation without changing both.

In our experiments, we rely on an open-source library called "segmentation models PyTorch"[2]which provides implementations of network architectures for attention segmentation and pre-trained encoders used in our experiments. The "segmentation models PyTorch" library provides the possibility of changing encoders for the networks for attention segmentation.

5.3 Model training

In the experiments described in chapter 6, deep learning models are trained on two different datasets. One dataset contains only manually labeled images, and the other contains additional labelling generated through Mixture-Gaussian-based segmentation. Some hyper-parameters are changed as part of the experiment procedure, as discussed in chapter 6, while some hyper-parameters remain static through all model training.

5.3.1 Training data

For all training data in all experiments the following data augmentation is applied; rotation (0°-35°), horizontal flip and vertical flip. The data augmentation is applied to both training images and ground truth segmentation maps.

For further data augmentation, Mixture Gaussian-based segmentation (MOG2) is used to generate additional labelling for training images. This acts a special case of data augmentation. As described in chapter 6, most experiments are carried out without the use of MOG2 for data augmentation. As an additional step in the experiments, certain models are trained on an enlarged dataset, where MOG2 is used to generate additional training data. The performance of these models is compared to performance of models trained on the original dataset.

5.3.2 hyper-parameters

For all training of deep learning models for attention segmentation in this thesis, the learning rate is set to 0.0001, the batch size is set to 4, and the Adam optimizer is used for optimization of the loss function.

[2]The "segmentation models PyTorch" library is available at `https://github.com/qubvel/segmentation_models.pytorch`.

5.4 Post-processing

The segmentation map predictions made by trained models have resolution $512x512$. As a post-processing step, we convert the predicted segmentation maps to the original 2448×2050 dimensions.

5.5 Hardware specifications

We train and test networks for attention segmentation in Google Colab. Colab is a free online service offering cloud based computing. Colab reserves a GPU for the user at each session, and the GPU specification can vary from session to session. We train networks using different GPUs provided by Colab. We do all testing of run-times of models in a single session, with hardware specifications as listed.

- **CPU:** Intel Xeon 2.00 Ghz
- **RAM:** 12GB
- **GPU:** Tesla T4

5.6 Implementation details for Mixture-Gaussian-based segmentation

We generate additional segmentation map labelling for training data by using Mixture-Gaussian-based segmentation (MOG2). For this, we rely on the OpenCV-Python library, which is a large library designed to solve computer vision tasks. We use the OpenCV MOG2 implementation to generate the additional labelling and we use the OpenCV library implementations of dilation and erosion to remove noise in the segmentation maps. The pipeline for creating labels using MOG2 is shown in figure 5.2. The pipeline starts with input images presented as time series to the MOG2 algorithm. Images presented as time series refers to images sorted after time of capture. Segmentation maps are created from MOG2, however, these segmentation maps are noisy with many irrelevant dust particles labelled foreground. To remove noise in the segmentation maps, 10×10 dilation is followed by 20×20 erosion. Dilation removes "holes" in the segmentation maps, by labelling pixels foreground if surrounding pixels are labelled foreground. Erosion removes noise by removing small particles. If a pixel labelled foreground does not have a sufficient amount of surrounding foreground particles, it is labelled background through erosion.

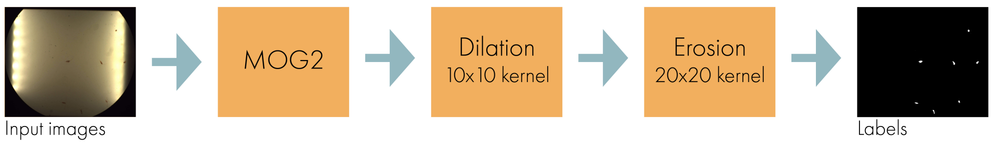


Figure 5.2: Framwerok for generating additional labelling with MOG2

Chapter 6

Experiments

In our experiments, we divide the data into a training set and a validation set. In the training set we place 284 images, and in the validation set we place 28 images. We first conduct experiments on the original dataset that contains only manually labeled images. We then conduct some experiments on a dataset that is enlarged with additional training images labeled through Gaussian-Mixture-based segmentation. We use the same validation set with only manually labelled data in all experiments. In this chapter we refer to networks for attention segmentation as "networks". The term "network-encoder pairing" refers to pre-trained encoders used as backbone for networks for attention segmentation. We use encoders pre-trained on ImageNet in all our experiments.

6.1 Deep learning attention segmentation with manually labeled data

The aim of this thesis is to find a suitable method for real-time in-situ attention segmentation of planktonic images. The method should be fast enough for real-time applications while at the same time yielding high segmentation accuracy. Through a process of elimination, we narrow down candidate methods. We perform initial experiments using five selected deep neural networks and employ the Resnet-101 encoder for all networks. These networks are U-net, U-net++, LinkNet, Deeplab V3+, and Pyramid attention network (PAN). Based on results from the initial experiments we select two network architectures for further experiments. We employ five different encoders, ResNet-101, ResNet-34, EfficientNet b-0, EfficientNet b-7, and MobileNet v2, for both selected network architectures. Finally, we choose a subset of the network-encoder pairings for training over an increased number of epochs.

We evaluate models by calculating dice score, precision, and recall for predictions on the validation set. In addition, we measure prediction run times for all models. Prediction run times are found by measuring the time for prediction of one image. In order to find the best method for attention segmentation on our planktonic dataset we perform the following steps.

1. We find a suitable loss function for use in training
2. We implement and train a selection of network architectures with ResNet-101 encoder over 15 epochs
3. We train two selected networks with 5 different encoders over 15 epochs to see if we can obtain faster run times and increased segmentation accuracy by using different encoders
4. We train two selected networks with selected encoders over 40 epochs

In steps 1, 2, and 3 we train the networks over 15 epochs. In step 4 we train over 40 epochs. Ideally we would have trained all networks, and all network-encoder pairings over 40 epochs in steps 1, 2, and 3. However, since training is time demanding, we choose the network-encoder pairings achieving the best result in the initial steps for training over 40 epochs in step 4.

Step 1: Finding a suitable loss function

There exist no single loss function that is ideal for all segmentation tasks, and we therefore test different loss functions for our segmentation task. We train U-net++ over 15 epochs using different loss functions. The loss functions we test are BCE, weighted BCE with positive weights set to 2, 7, and 15, and focal loss with γ set to 1, 2, and 3. BCE is commonly used in machine learning classification tasks with balanced datasets. Weighted BCE prioritizes positive weights, and is therefore useful for machine learning tasks with imbalanced datasets. Focal loss is designed for segmentation tasks with imbalanced datasets and prioritizes hard-to-segment samples.

Step 2: Finding the best networks for attention segmentation on our planktonic dataset

In step 1 we observe that U-net++ yields the highest dice score when we use BCE 2 in training. We therefore use BCE 2 in training when testing the performance of U-net, LinkNet, Deeplab V3+, and Pyramid attention network. However, we observe in initial testing that these networks have significantly lower dice coefficient than the U-net++ when the BCE 2 loss function is used in training. We perform further experiments by training all networks over 15 epochs with the BCE 7 loss function and obtain improved results for these networks doing so.

Step 3: Finding the best encoders for selected networks

Based on the experiments in sections 1 and 2 we select two networks for further training. We train the two selected networks over 15 epochs with 5 different encoders. This is done to see if we can obtain improved segmentation and lower run times by employing different encoders.

Step 4: Training selected network-encoder pairings over 40 epochs

Based on experiments in sections 1, 2, and 3, we select network-encoder pairings for additional training over 40 epochs. Step 4 is the final step towards finding a suitable deep learning model for attention segmentation on the planktonic dataset. We use the results obtained in this step to select our final recommendation of deep learning model for attention segmentation for our use case.

6.2 Training deep learning models on an enlarged dataset

We create a larger dataset by generating segmentation map labelling for training images using the MOG2 method followed by dilation and erosion, as discussed in chapter 4. We train selected models over 40 epochs on the enlarged training set. The models used are the ones that obtained the best results when trained on the dataset containing only manually labeled data. We finally compare performance of models trained on our original dataset, with performance of models trained on the enlarged dataset.

Chapter 7

Results

In this chapter we present our results. We first present results for all steps in the process of finding the best model for attention segmentation on the original manually labelled dataset. Finally, we compare results for models trained on the original dataset and models trained on the enlarged dataset. For evaluation of model performance we give most emphasis to the dice score.

7.1 Training networks on the original dataset

In this section we present results for models trained on the original dataset. We first test loss functions to find a suitable loss function for the segmentation problem. We then test several models, and choose the best models at each step for further experiments. Finally we train selected models over an additional number of epochs, to obtain our conclusive results for models trained on the original dataset.

7.1.1 Finding a suitable loss function

We train U-net++ over 15 epochs using different loss functions. We use a ResNet-101 pre-trained encoder as backbone for the U-net++. The loss functions used are BCE, weighted BCE with positive weights set to 2,7,and 15, and focal loss with $\alpha = 1$ and γ set to 1,2,and 3. We refer to the loss functions with the following abbreviations.

- BCE = Binary cross entropy loss
- BCE 2 = Binary cross entropy loss with positive weights set to 2
- BCE 7 = Binary cross entropy loss with positive weights set to 7
- BCE 15 = Binary cross entropy loss with positive weights set to 15
- Focal g1 = Focal loss with γ set to 1 and α set to 1
- Focal g2 = Focal loss with γ set to 2 and α set to 1
- Focal g3 = Focal loss with γ set to 3 and α set to 1

Loss function	Dice	Precision	Recall
BCE	0.7802	0.8286	0.7383
BCE 2	0.8142	0.7837	0.8493
BCE 7	0.7944	0.7112	0.9028
BCE 15	0.7697	0.6642	0.9181
Focal g1	0.8024	0.8229	0.7846
Focal g2	0.7980	0.8422	0.7597
Focal g3	0.7913	0.8450	0.7455

Table 7.1: Performance of U-net++ using different loss functions. 15 epochs

We see in table 7.1 that for the U-net++ the best dice score is achieved when using the BCE 2 loss function. This indicates that out of the tested loss functions, BCE 2 is most suitable for the segmentation task. Thus, we use the BCE 2 loss function in further experiments.

7.1.2 Testing all networks with ResNet-101 encoder

We train U-net, U-net++, LinkNet, DeepLab V3+, and Pyramid attention network over 15 epochs with a ResNet-101 encoder as backbone. We first train all networks using the BCE 2 loss function. However, in further experiments, we observe that U-net, LinkNet and DeepLab V3+ achieve a higher dice score when the BCE 7 loss function is used. This suggests that the when choosing a loss function we should consider not only the segmentation task at hand, but also the network architecture of models using the loss function. In table 7.2 we show results from the networks trained with the BCE 2 loss function and the BCE 7 loss function. In addition, we show prediction runtime, which is the time it takes for the different networks to predict the segmentation map for one image. The choice of loss function does not influence complexity of models or the runtime for predictions.

We read from table 7.2 that the best dice score is achieved with U-net++ using the BCE 2 loss function, and we therefore select the U-net++ for further experiments. One of the research questions in this thesis focuses on finding a model that provides good segmentation performance and fast runtimes for predictions. We see that the LinkNet trained using the BCE 7 loss function achieves the second highest dice score, and one of the fastest prediction runtimes. We therefore also select the LinkNet for further experiments.

7.1.3 Testing U-net++ and LinkNet with different encoders

We test 5 different encoders as backbones for U-net++ and LinkNet to see if this can lead to improvement in segmentation performance and run time for predictions. We use the BCE 2 loss function for U-net++ and the BCE 7 loss function for LinkNet. In table 7.3 we show results for U-net++, and in table 7.4 we show res-

Network architecture and loss function	Dice	Precision	Recall	pred runtime
U-net BCE 2 U-net BCE 7	0.7755 0.7909	0.7704 0.7221	0.7821 0.8768	1.5230s
U-net++ BCE 2 U-net++ BCE 7	0.8142 0.7944	0.7837 0.7112	0.8493 0.9028	5.6116s
LinkNet BCE 2 LinkNet BCE 7	0.7441 0.7971	0.7711 0.7405	0.7204 0.8653	1.3406s
DeepLab V3+ BCE 2 DeepLab V3+ BCE 7	0.7268 0.7441	0.7437 0.6558	0.7132 0.8610	1.6832s
PAN BCE 2 PAN BCE 7	0.7574 0.7515	0.7589 0.6631	0.7575 0.8696	1.2705s

Table 7.2: Performance metrics of the networks trained using BCE 2 and BCE 7 loss functions. 15 epochs

Encoder	Dice	Precision	Recall	pred runtime
Resnet-101	0.8142	0.7837	0.8493	5.6116s
Resnet-34	0.8121	0.7838	0.8448	2.1368s
Efficient-net b-0	0.7975	0.8019	0.7951	1.2900s
Efficient-net b-7	0.8207	0.8122	0.8314	4.8142s
Mobilenet V2	0.7379	0.7498	0.7300	0.9870s

Table 7.3: Unet++; Performance metrics using different encoders. 15 epochs

ults for LinkNet. We observe that the best dice score is achieved for U-net++ with EfficientNet b-7 encoder. The fastest runtimes are achieved with LinkNet and Mobilenet V2 encoder. We observe that the runtimes for LinkNet with ResNet-34 and EfficientNet b-0 are also low, and that the dice score is higher with these encoders than with Mobilenet V2. We select U-net ++ with EfficientNet b-7, and LinkNet with ResNet-34, EfficientNet b-0 and Mobilenet V2 for further experiments.

7.1.4 Testing U-net++ and LinkNet with selected encoders over an increased number of epochs

We train U-net++ and LinkNet over 40 epochs to see if this yields improved segmentation performance. We use U-net++ with EfficientNet b-7 encoder, and

Encoder	Dice	Precision	Recall	pred runtime
Resnet-101	0.7971	0.7405	0.8653	1.3406s
Resnet-34	0.6884	0.5849	0.8385	0.6335s
Efficient-net b-0	0.6746	0.6114	0.7554	0.5952s
Efficient-net b-7	0.7950	0.7124	0.9017	3.4208s
Mobilenet V2	0.6165	0.4705	0.8960	0.4375s

Table 7.4: Linknet; performance metrics using different encoders. 15 epochs

Network and encoder	Dice	Precision	Recall	pred runtime
U-net++ w. EfficientNet b-7	0.8324	0.8178	0.8504	4.8142s
LinkNet w. ResNet-34	0.8068	0.7907	0.8255	0.6335s
LinkNet w. EfficientNet b-0	0.7688	0.6881	0.8743	0.5952s
LinkNet w. Mobilenet V2	0.7311	0.6076	0.9203	0.4375s

Table 7.5: Performance over 40 epochs for U-net++ and LinkNet with selected encoders

LinkNet with ResNet-34, EfficientNet b-0 and Mobilenet V2. We use the BCE 2 loss function for U-net++ and the BCE 7 loss function for LinkNet. The results are shown in table 7.5. Runtime does not change by training networks over additional epochs, as this does not influence model complexity, so runtimes for the models trained over 40 epochs are the same as runtimes for models trained over 15 epochs.

We observe in table 7.5 that the highest dice score (0.8324) for all experiments is achieved for U-net++ with EfficientNet b-7 encoder. Examples of resulting segmentation maps for U-net++ with EfficientNet b-7 are shown in figure 7.1 We also observe that LinkNet with ResNet-34 encoder achieves a low prediction run time, and a high dice score (0.8068). Examples of resulting segmentation maps for LinkNet with ResNet-34 are shown in figure 7.2.

7.2 Training networks on the enlarged dataset

We train selected models on the enlarged dataset over 40 epochs and compare results with models trained on the original dataset. We use U-net++ with EfficientNet b-7 and LinkNet with ResNet-34. For the LinkNet, we use two different loss functions in training, BCE 2 and BCE 7, to see if training over a larger number of epochs has an influence on the results with different loss functions used in training.

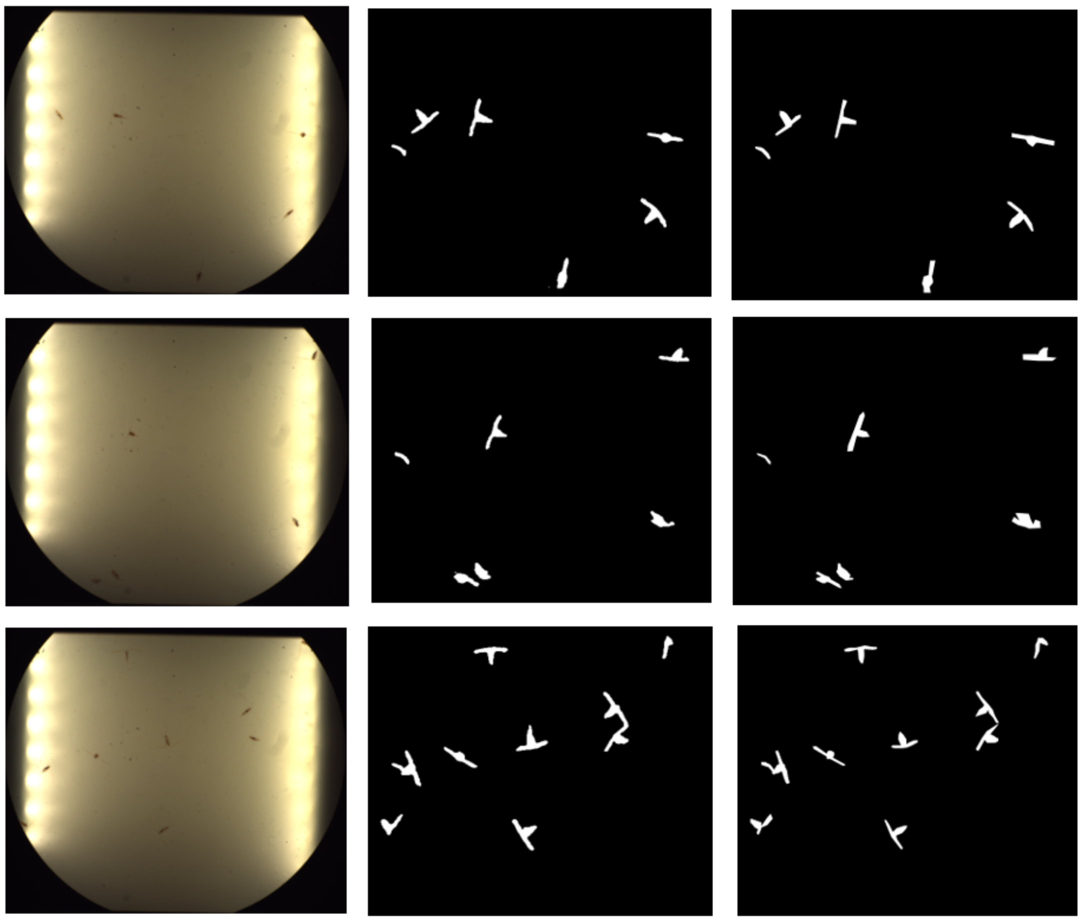


Figure 7.1: Segmentation results from U-net++ with EfficientNet b-7. To the left are input images, in the middle predicted segmentation maps, and to the right are ground truth segmentation maps

The results for U-net++ are shown in table 7.6, and in figure 7.3 are examples of predicted output segmentation maps from U-net++ trained on the original dataset, and the enlarged dataset. We observe in table 7.6 that the dice score is almost identical in the experiments. However we observe that precision is higher, and recall is lower for the model trained on the enlarged dataset. This indicates that the number of true negatives is larger, and the number of true positives is smaller for the model trained on the enlarged dataset. This is confirmed from the examples in figure 7.3 where we observe that the regions labeled foreground are slightly smaller in predictions made by the model trained on the enlarged dataset.

The results for LinkNet are shown in table 7.7, and in figure 7.4 are examples of predicted output segmentation maps from LinkNet trained on the original dataset and the enlarged dataset using the BCE 7 loss function. We observe in table

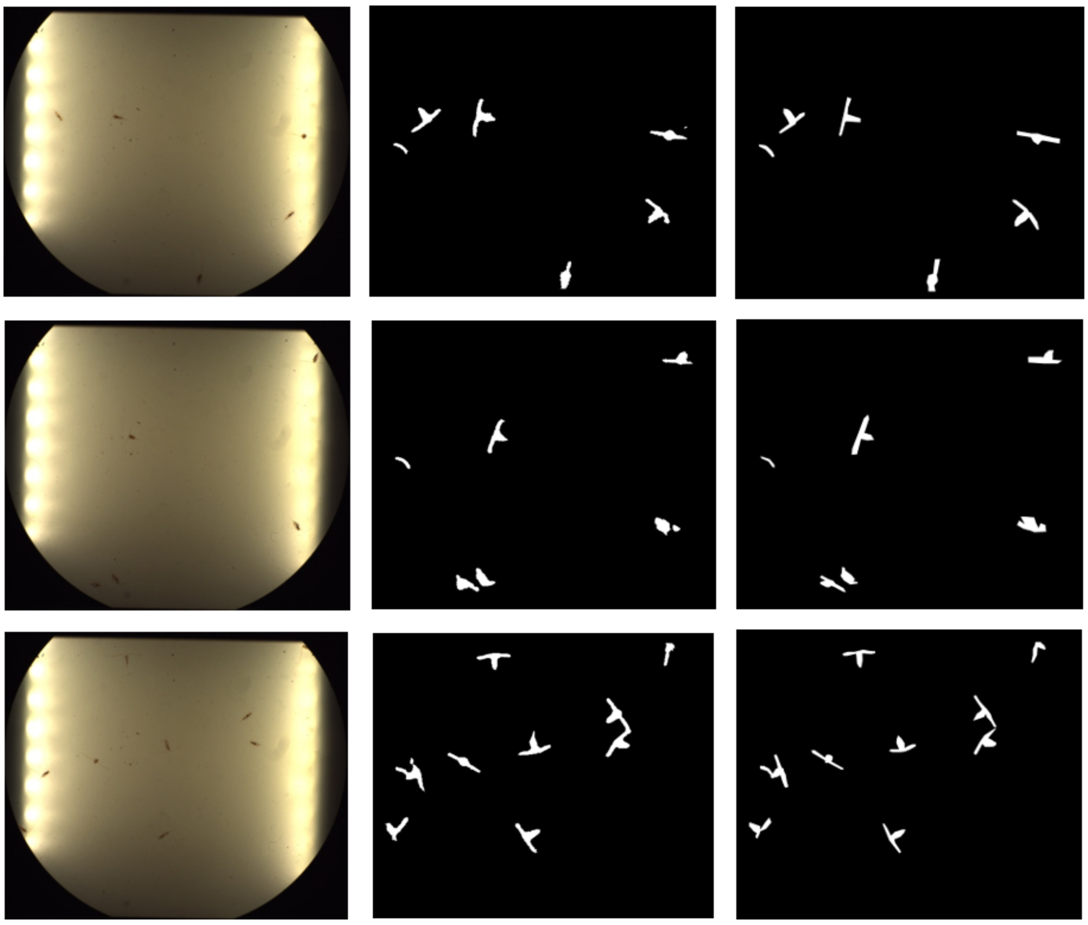


Figure 7.2: Segmentation results from LinkNet with ResNet 34. To the left are input images, in the middle predicted segmentation maps, and to the right are ground truth segmentation maps

7.7 that all evaluation metrics are very similar for the different datasets when the BCE 7 loss function is used in training. This is confirmed from the examples in figure 7.4 where we see that segmentation maps for the LinkNet models trained on the two different datasets are very similar. We observe that when the models are trained with the BCE 2 loss function, dice score and precision is larger for the model trained on the enlarged dataset, while recall is similar. We see that when training over a larger number of epochs we get slightly better results using the BCE 7 loss function than the BCE 2 loss function. However the difference in performance from using the different loss functions is smaller when LinkNet is trained over 40 epochs than when it is trained over 15 epochs.

Training data	Dice	Precision	Recall
Original dataset	0.8324	0.8178	0.8504
Enlarged dataset	0.8314	0.8713	0.7965

Table 7.6: Unet++ with EfficientNet b-7; trained over 40 epochs on original dataset and enlarged dataset

Training data and Loss function	Dice	Precision	Recall
Original dataset BCE 2	0.7865	0.8016	0.7740
Original dataset BCE 7	0.8068	0.7907	0.8255
Enlarged dataset BCE 2	0.8004	0.8348	0.7704
Enlarged dataset BCE 7	0.8085	0.7903	0.8297

Table 7.7: Linknet w. ResNet-34; trained over 40 epochs with BCE 2 and BCE 7 loss functions on original dataset and enlarged dataset

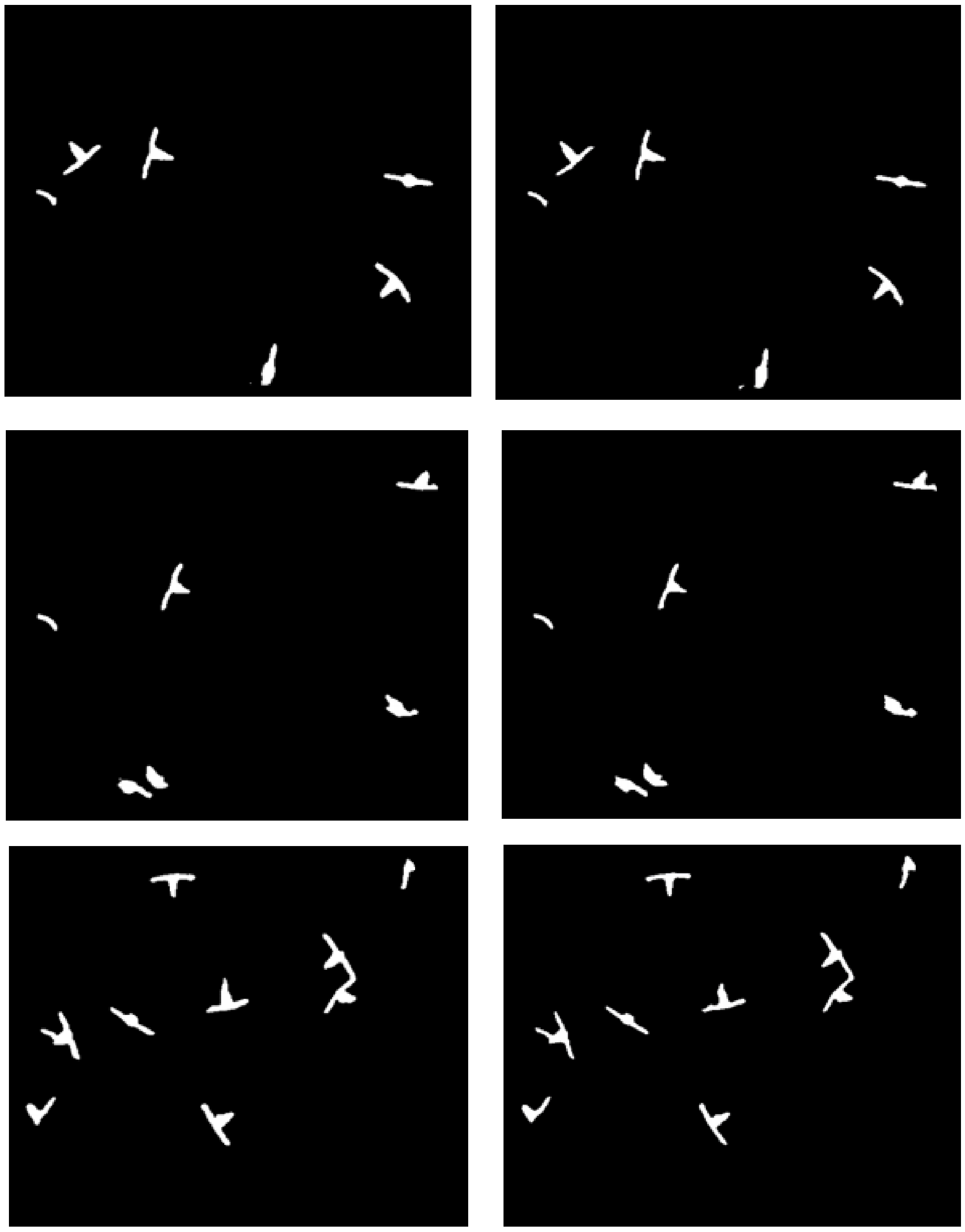


Figure 7.3: Segmentation results from U-net++ with EfficientNet b-7. To the left are predictions from the model trained on the original dataset, and to the right are predictions from the model trained on the enlarged dataset

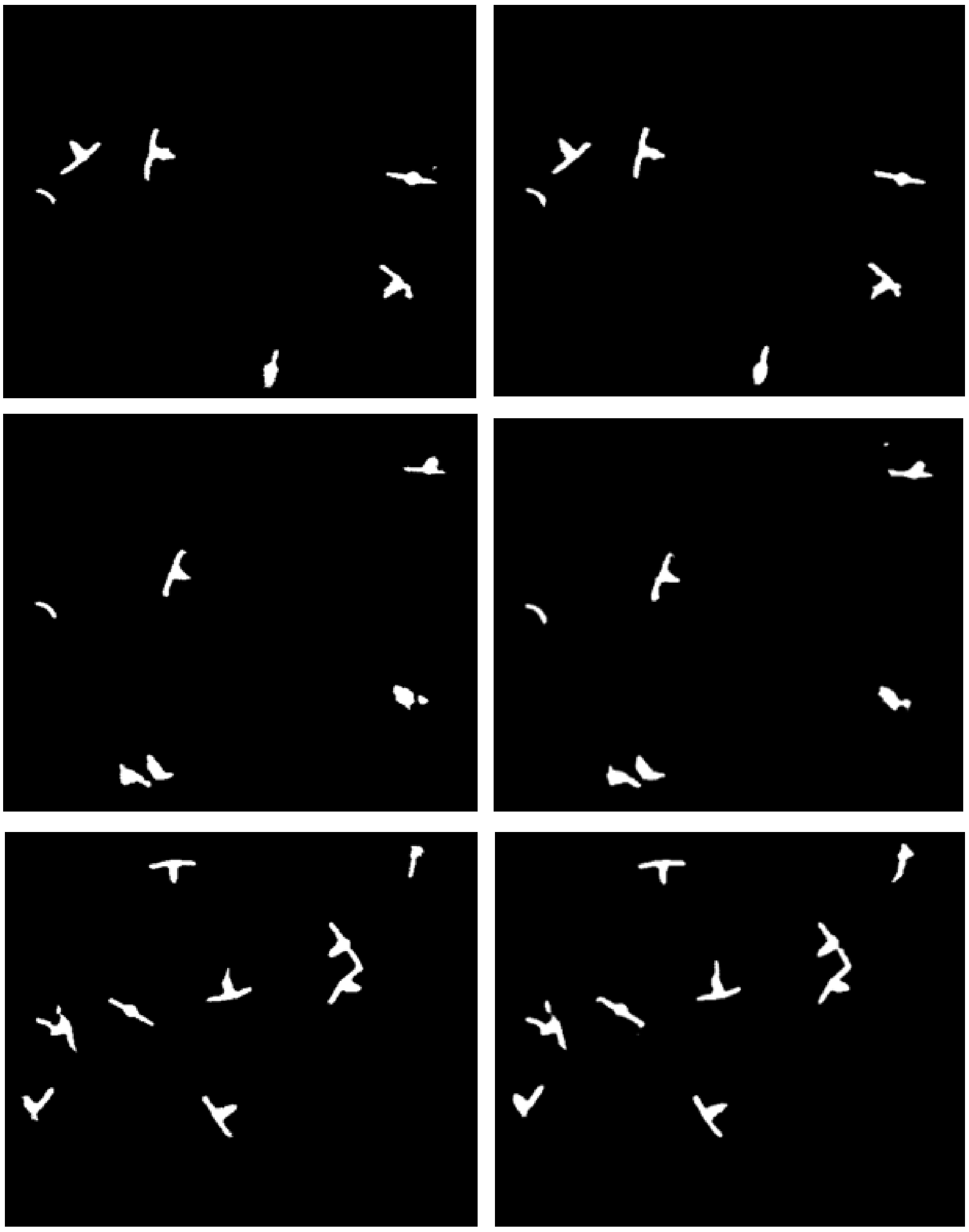


Figure 7.4: Segmentation results from LinkNet with ResNet-34 with the BCE 7 loss function used in training. To the left are predictions from the model trained on the original dataset, and to the right are predictions from the model trained on the enlarged dataset

Chapter 8

Discussion

This chapter answers and reflects on the research questions presented in chapter 1. The discussion is based on the results obtained in the experiments elaborated in chapter 6.

8.1 The best performance achieved

In our experiments, we achieve the highest dice score (0.8324) for our segmentation task by using U-net++ with EfficientNet b-7 encoder. The dice coefficient measures the similarity between predictions and ground truth segmentation maps in the validation set. The model achieving the highest dice score has the most similarity between predictions and ground truth segmentation maps. Thus, the model achieving the highest dice score in the experiments described in chapter 6 is consider the best performing model.

8.2 The best deep learning model for real-time attention segmentation

In our experiments, we achieve a high dice score (0.8068) and low runtime for prediction of one image (0.6335s) by using LinkNet with ResNet-34 encoder. The lowest prediction runtime (0.4375s) is obtained from LinkNet with MobileNet V2 encoder. However for this model, the dice score is only 0.7311, and we believe that for most tasks, this significant decrease in dice score is not justified by the decreased prediction runtime. Thus we conclude from our experiments that the best deep learning model for real-time attention segmentation on our planktonic dataset is LinkNet with ResNet-34 encoder.

8.3 The suitable loss function for the segmentation task

In [23], the process of choosing a suitable loss function for image segmentation is discussed in the context of the segmentation task at hand. We observe from our results that the most suitable loss function was different for the different network architectures, even though all were trained on the same dataset with the objective of solving the same segmentation task. We tested several different loss functions for U-net++ and obtained the best results when the BCE 2 loss function was used. However, we observed that for U-net, DeepLab V3+, and LinkNet, the models achieved higher dice scores when the BCE 7 loss function was used in training. We also saw that when training networks over 40 epochs the difference in performance of LinkNet trained with the BCE 2 and BCE 7 loss functions decreased. This indicates that choosing the correct loss function can give faster convergence, but that the difference in performance from using different loss function can decrease when models are trained over a larger number of epochs. Our experiments indicate that the choice of loss function effects model performance, especially when models are trained over a small number of epochs. Furthermore, our experiments indicate that for our segmentation problem the most suitable loss function for U-net++ is BCE 2, and the most suitable loss function for LinkNet is BCE 7.

8.4 The effect of adding MOG to the training process

The dice scores for models trained on our original dataset, and the enlarged dataset are very similar. For the U-net++ we see that precision is higher and recall is lower for the model trained on the enlarged dataset. As discussed in chapter 4, the labelling in the original dataset is not perfectly accurate, and the regions labeled foreground are larger than the actual objects they represent. All evaluation metrics are calculated by comparing predicted segmentation maps to manually labelled segmentation maps in the validation set. Therefore, a perfect model for attention segmentation would have close to a 100 percent precision score, indicating that all pixels predicted as foreground are also labelled as foreground. In addition, it would have a slightly lower recall score indicating that the predicted foreground segmentation map includes fewer pixels than the foreground segmentation map in the labelling. Thus, we conclude that for the U-net++, the model trained on the enlarged dataset has slightly better performance than the model trained on our original dataset. In figure 7.3 we can see that the model trained on the larger dataset has slightly smaller regions representing foreground. For the LinkNet models, we see that the models trained on the different datasets using the BCE 7 loss function have very similar results on all evaluation metrics. However, we see that for the LinkNet models trained using the BCE 2 loss function, dice score and precision is higher for the model trained on the enlarged dataset.

All together, these results indicate that a small increase in segmentation performance is obtained when training models on the enlarged dataset containing images labelled using MOG. As discussed in chapter 4, the regions labelled foreground in the MOG labelled segmentation maps are smaller than the objects they represent. This is opposite to the original dataset. Therefore, one reason for the slight improvement in segmentation performance could be that the MOG labelled images and manually labeled images balance out each others weaknesses to some degree. Another reason could be that models are able to generalize better when more training data is available.

Chapter 9

Conclusion

In this thesis, methods for real-time in-situ attention segmentation in images captured by an autonomous underwater vehicle (AUV), were investigated. The goal was to find a method that provides low run-times and high segmentation performance to improve classification and detection of plankton in the AUV. The AUV is used to detect and classify plankton, because insight on changes in abundance and distribution of plankton can help scientist better understand effects of climate change on aquatic life.

Attention segmentation can be used to localize relevant information in the planktonic images, and it can act as a backbone for classification of plankton. The experiments in this thesis were aimed at answering research questions defined in chapter 1.

1. The best performing model for attention segmentation for the planktonic datasets was found to be U-net++ with a pre-trained EfficientNet b-7 encoder as backbone. This model obtained a dice score of 0.8324. For evaluation of segmentation performance, dice coefficient was given the most emphasis. In addition precision and recall were used as evaluation metrics, to give further insight in the relative distribution between positives and negatives in the predicted segmentation maps.

2. For real time applications a compromise of lower performance and faster run times often has to be made. The best model for real-time applications was found to be LinkNet with a pretrained ResNet-34 encoder as backbone. This model yielded low run-times for predictions, and a high dice score of 0.8068.

3. In experiments, the most suitable loss function for U-net++ was found to be binary cross entropy with positive weights set to 2, while for LinkNet, the most suitable loss function was found to be binary cross entropy with positive weights set to 7.

4. Finally, it was found that training networks on a dataset enlarged through Mixture-Gaussian-based segmentation for images in the training set gave a small increase in segmentation performance. The Mixture-Gaussian-based segmentation acted as a special case of data augmentation.

A wide array of deep learning methods for attention segmentation were implemented and tested for a dataset with images containing planktonic organisms. Several network architectures for attention segmentation were tested. As backbones for networks for attention segmentation, several pretrained encoders were used. In addition, different loss functions were tested in the training process. Through a process of elimination, candidate methods were narrowed down. Finally two models were recommended for attention segmentation for the planktonic images. The first model achieved the best segmentation performance in experiments, and the second model provided low run-times and high performance in experiments. Furthermore, Mixture-Gaussian-based segmentation was used to generate additional labelling for training data, acting as a special case of data augmentation. Models for attention segmentation were trained on the original dataset containing only manually labeled images, and on an enlarged dataset containing both manually labelled images and images labelled using Mixture-Gaussian-based segmentation. Finally, a comparison of performance of models trained on the two different datasets was conducted.

Chapter 10

Future work

In this thesis many deep learning models for attention segmentation were implemented and tested, and several loss functions were used in training of models. Furthermore, models were trained on a larger dataset with additional data generated through Mixture-Gaussian-based segmentation. In future work, it would be interesting to widen the scope of candidate methods, to see if testing even more methods for attention segmentation could yield even better segmentation and faster run-times for predictions. Further experiments that are considered interesting for future work are listed and discussed below.

1. There are many deep learning models for attention segmentation available in the literature that were not implemented in this thesis. In future work we would like to test more models for attention segmentation for our planktonic dataset, to see if this can lead to improved segmentation.

2. We tested a variation of pre-trained encoders as backbones for deep learning models for attention segmentation. However many more encoders are available, and in future work we would like to test more encoders to see if this can yield improved segmentation results and faster run-times for predictions.

3. Several different loss functions were tested with U-net++ in experiments. However, we observed that the best choice of loss function was not only dependant on the segmentation task, but also the network architecture of models. Thus, we would like to perform a more comprehensive test of loss functions for all models. In addition, we would like to further test different parameter tuning for the loss functions.

4. In this thesis, we used Mixture-Gaussian-based segmentation (MOG) to generate additional labelling for images in the training dataset. These images were already manually labeled, and this acted as a special kind of data augmentation. However, we have an abundance of unlabeled images, and in future work we would like to generate additional labelling using MOG for

images that are not already manually labeled. This way we can obtain an enlarged dataset with new images, where some images are manually labeled and some images are labeled using MOG, and use this enlarged dataset in training of deep neural networks for attention segmentation.

Bibliography

[1] M. Winder and U. Sommer, ‘Phytoplankton response to a changing climate,’ *Hydrobiologia*, vol. 698, no. 1, pp. 5–16, 2012.

[2] A. J. Richardson, ‘In hot water: Zooplankton and climate change,’ *ICES Journal of Marine Science*, vol. 65, no. 3, pp. 279–295, 2008.

[3] G. C. Hays, A. J. Richardson and C. Robinson, ‘Climate change and marine plankton,’ *Trends in ecology & evolution*, vol. 20, no. 6, pp. 337–344, 2005.

[4] A. Saad, A. Stahl, A. Våge, E. Davies, T. Nordam, N. Aberle, M. Ludvigsen, G. Johnsen, J. Sousa and K. Rajan, ‘Advancing ocean observation with an ai-driven mobile robotic explorer,’ *Oceanography*, vol. 33, no. 3, pp. 50–59, 2020.

[5] P. Kim, ‘Convolutional neural network,’ in *MATLAB deep learning*, Springer, 2017, pp. 121–147.

[6] Z. Zivkovic, ‘Improved adaptive gaussian mixture model for background subtraction,’ in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, vol. 2, 2004, pp. 28–31.

[7] Z. Zivkovic and F. Van Der Heijden, ‘Efficient adaptive density estimation per image pixel for the task of background subtraction,’ *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, 2006.

[8] S. Bergum, ‘Ttk4555 msc thesis object detection and instance segmentation of planktonic organisms using mask r cnn for real time in situ image processing,’ Jun. 2020.

[9] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz and D. Terzopoulos, ‘Image segmentation using deep learning: A survey,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[10] D. Kaur and Y. Kaur, ‘Various image segmentation techniques: A review,’ *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 5, pp. 809–814, 2014.

[11] S. Yuheng and Y. Hao, ‘Image segmentation algorithms overview,’ *arXiv preprint arXiv:1707.02051*, 2017.

[12] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou and G. Cottrell, 'Understanding convolution for semantic segmentation,' in *2018 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2018, pp. 1451–1460.

[13] I. Goodfellow, Y. Bengio, A. Courville and Y. Bengio, *Deep learning*, 2. MIT press Cambridge, 2016, vol. 1.

[14] M. Mohri, A. Rostamizadeh and A. Talwalkar, *Foundations of machine learning*. 2018.

[15] L. Deng and D. Yu, 'Deep learning: Methods and applications,' *Foundations and trends in signal processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[16] F. Bre, J. M. Gimenez and V. D. Fachinotti, 'Prediction of wind pressure coefficients on building surfaces using artificial neural networks,' *Energy and Buildings*, vol. 158, pp. 1429–1441, 2018.

[17] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, 2015, vol. 2018.

[18] S. Sharma and S. Sharma, 'Activation functions in neural networks,' *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[19] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, 'Activation functions: Comparison of trends in practice and research for deep learning,' *arXiv preprint arXiv:1811.03378*, 2018.

[20] Y. Bengio, P. Simard and P. Frasconi, 'Learning long-term dependencies with gradient descent is difficult,' *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[21] Z. Hu, J. Zhang and Y. Ge, 'Handling vanishing gradient problem using artificial derivative,' *IEEE Access*, vol. 9, pp. 22 371–22 377, 2021.

[22] P. Kim, 'Matlab deep learning,' *With machine learning, neural networks and artificial intelligence*, vol. 130, p. 21, 2017.

[23] S. Jadon, 'A survey of loss functions for semantic segmentation,' in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, IEEE, 2020, pp. 1–7.

[24] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[25] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana and A. Verri, 'Are loss functions all the same?' *Neural computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[26] P. J. Huber, 'Robust estimation of a location parameter,' in *Breakthroughs in statistics*, Springer, 1992, pp. 492–518.

[27] S. Kullback and R. A. Leibler, 'On information and sufficiency,' *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[28] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization,' *arXiv preprint arXiv:1412.6980*, 2014.

[29] T. Dietterich, 'Overfitting and undercomputing in machine learning,' *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[30] C. Shorten and T. M. Khoshgoftaar, 'A survey on image data augmentation for deep learning,' *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting,' *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] S. Ioffe and C. Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift,' in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[33] M. M. Palatucci, D. A. Pomerleau, G. E. Hinton and T. Mitchell, 'Zero-shot learning with semantic output codes,' 2009.

[34] Y. Xian, C. H. Lampert, B. Schiele and Z. Akata, 'Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly,' *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[35] S. J. Pan and Q. Yang, 'A survey on transfer learning,' *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'Imagenet: A large-scale hierarchical image database,' in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[37] L. Taylor and G. Nitschke, 'Improving deep learning using generic data augmentation,' *arXiv preprint arXiv:1708.06020*, 2017.

[38] H. Inoue, 'Data augmentation by pairing samples for images classification,' *arXiv preprint arXiv:1801.02929*, 2018.

[39] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, 'Recent advances in convolutional neural networks,' *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[40] R. Yamashita, M. Nishio, R. K. G. Do and K. Togashi, 'Convolutional neural networks: An overview and application in radiology,' *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[41] J. Long, E. Shelhamer and T. Darrell, 'Fully convolutional networks for semantic segmentation,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[42] 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, 2015, pp. 234–241.

[43] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh and J. Liang, 'Unet++: A nested u-net architecture for medical image segmentation,' in *Deep learning in medical image analysis and multimodal learning for clinical decision support*, Springer, 2018, pp. 3–11.

[44] 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*, 2017, pp. 4700–4708.

[45] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang and Z. Tu, 'Deeply-supervised nets,' in *Artificial intelligence and statistics*, PMLR, 2015, pp. 562–570.

[46] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff and H. Adam, 'Encoder-decoder with atrous separable convolution for semantic image segmentation,' in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[47] 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 and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

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

[49] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, 'The cityscapes dataset for semantic urban scene understanding,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[50] H. Li, P. Xiong, J. An and L. Wang, 'Pyramid attention network for semantic segmentation,' *arXiv preprint arXiv:1805.10180*, 2018.

[51] A. Chaurasia and E. Culurciello, 'Linknet: Exploiting encoder representations for efficient semantic segmentation,' in *2017 IEEE Visual Communications and Image Processing (VCIP)*, IEEE, 2017, pp. 1–4.

[52] 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*, 2016, pp. 770–778.

[53] D. Chen, F. Hu, G. Nian and T. Yang, 'Deep residual learning for nonlinear regression,' *Entropy*, vol. 22, no. 2, p. 193, 2020.

[54] T. Liu, M. Chen, M. Zhou, S. S. Du, E. Zhou and T. Zhao, 'Towards understanding the importance of shortcut connections in residual networks,' *arXiv preprint arXiv:1909.04653*, 2019.

[55] M. Tan and Q. Le, 'Efficientnet: Rethinking model scaling for convolutional neural networks,' in *International Conference on Machine Learning*, PMLR, 2019, pp. 6105–6114.

[56] 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.

[57] V. Pihur, S. Datta and S. Datta, 'Weighted rank aggregation of cluster validation measures: A monte carlo cross-entropy approach,' *Bioinformatics*, vol. 23, no. 13, pp. 1607–1615, 2007.

[58] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, 'Focal loss for dense object detection,' in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[59] M. Hossin and M. Sulaiman, 'A review on evaluation metrics for data classification evaluations,' *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[60] N. (https://stats.stackexchange.com/users/124257/nico), *F1/dice-score vs iou*, Cross Validated, URL:https://stats.stackexchange.com/q/488098 (version: 2020-09-18). eprint: `https://stats.stackexchange.com/q/488098` . [Online]. Available: `https://stats.stackexchange.com/q/488098`.

[61] P. KaewTraKulPong and R. Bowden, 'An improved adaptive background mixture model for real-time tracking with shadow detection,' in *Video-based surveillance systems*, Springer, 2002, pp. 135–144.

[62] L. Marcomini and A. L. Cunha, 'A comparison between background modelling methods for vehicle segmentation in highway traffic videos,' *arXiv preprint arXiv:1810.02835*, 2018.

[63] M. L. Comer and E. J. Delp III, 'Morphological operations for color image processing,' *Journal of electronic imaging*, vol. 8, no. 3, pp. 279–289, 1999.

Appendix

The paper based on the work in this thesis and accepted by The 14th International Conference on Machine Vision (ICMV 2021) is attached in the appendix.

MOG: a background extraction approach for data augmentation of time-series images in deep learning segmentation

Jonas Nagell Borgersen[1], Aya Saad[1], Annette Stahl[1]

[1]Dept. of Engineering Cybernetics, Norwegian University of Science and Technology, NTNU, Trondheim, Norway

ABSTRACT

Image segmentation is one of the key components in systems performing computer vision recognition tasks. Various algorithms for image segmentation have been developed in the literature. Among them, more recently, deep learning algorithms have been remarkably successful in performing this task. A downside with deep neural networks for segmentation is that they require a large amount of labeled dataset for training. This prerequisite is one of the main reasons that led researchers to adopt data augmentation approaches in order to minimize manual labeling efforts while maintaining highly accurate results. This paper uses classical non-deep learning methods for background extraction to increase the size of the dataset used to train deep learning attention segmentation algorithms when images are presented as time-series to the model. The method presented adopts the Gaussian mixture-based (MOG2) foreground-background segmentation followed by dilation and erosion to create masks necessary to train the deep learning models. It is applied in the context of planktonic images captured in situ as time series. Various evaluation metrics and visual inspection are used to compare the performance of the deep learning algorithms. Experimental results show higher accuracy achieved by the deep learning algorithms for time-series image attention segmentation when the proposed data augmentation methodology is utilized to increase the training dataset.

Keywords: data augmentation, image analysis, deep learning, background extraction algorithms, in-situ plankton taxa classification, segmentation

1. INTRODUCTION

Image segmentation is a classical problem in computer vision. Proposed algorithms in the literature for image segmentation, such as in [1, 2], tend to assign each pixel in a given image a class. In recent years, deep convolutional neural networks for semantic segmentation have achieved improved segmentation accuracy over classical, non-deep-learning methods. The introduction of the Fully Convolutional Network (FCN) architecture [3] and the legendary U-net [4] showed great potential for deep neural networks in image segmentation. Several other network architectures such as the DeeplabV3+ [5] and the Unet++ [6], among many others, have since shown improved results over the U-net and the FCN for various segmentation tasks. However, there are certain drawbacks with the deep neural networks for semantic segmentation, e.g., the networks require a sizable training dataset of correctly labeled images to achieve satisfactory accurate results. Attempts for data augmentation were proposed in the literature to compensate for the small datasets used in the training process of the deep learning models. Those attempts generally exert some operations on the images provided in the dataset for training; such operations include geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, generative adversarial networks, neural style transfer, and meta-learning [7]. Still, data augmentation efforts do not consider the massive amount of unlabeled data captured during the data collection process, especially in the context when the data is provided as time-series.

The work presented in this paper aims at performing in-situ detection of planktonic data captured as time-series by a mobile platform, lightweight autonomous vehicle (AUV), described in [8]. The entire process of pixel-wise classification and detection of plankton could be done through instance segmentation [9, 10]. However, deep learning methods for instance segmentation detect planktonic organisms that belong to predefined classes manually labeled by domain experts. In real-life situations, the AUV may encounter thousands of planktonic species [11] that have never been seen nor included in the labeled dataset.

The contributions in this paper consist of implementing data augmentation in a novel manner, making use of the massive amount of the data captured in situ. At the same time, minimize the manual labeling efforts exerted by domain experts, which is time-consuming and subject to human error. The data augmentation method utilized adopts classical background extraction methods over images captured as time series to perform attention segmentation by extracting the foreground objects from the background. The output of this segmentation module consists of masks used to increase the dataset for the deep learning segmentation algorithm. The method presented applies the Gaussian mixture-based (MOG2) [12, 13] to perform foreground-background segmentation followed by dilation and erosion [14] to remove detected noise in the masks. The reason behind applying this method on the time series captured images is that such algorithms do not require labeled images for training. At the same time, they provide decent up-sizing to the dataset aiming at improving the accuracy of the deep learning segmentation model.

The deep learning semantic segmentation network architectures are trained on the datasets that contain both types of masks manually and automatically generated. The original manually labeled dataset consists of 312 planktonic image scenes with corresponding ground truth image masks labeled by domain experts, biologists. The manually labeled masks are not a perfect pixel-by-pixel representation of foreground and background; most of the extracted regions detected as foreground are larger than the actual object size due to human error. Furthermore, some planktonic organisms present in images were mistakenly not labeled. As the manual labeling is not entirely accurate, we do not expect any model to produce perfect pixel-by-pixel attention segmentation on the data.

This paper further presents a comparison between the segmentation results of networks trained on manually labeled data with a network trained on the augmented dataset using Gaussian mixture-based (MOG2) segmentation to confirm the usefulness of the novel augmentation approach. Empirical results show that the proposed data augmentation methodology improves the trained models accuracy.

The rest of the paper is organized as follows: Section 2 introduces some preliminary knowledge related to this paper. Section 3 presents related work in data augmentation. Section 4 explains the methodolgy. Section 5 presents the experimental results. Finally, section 6 concludes the paper and presents some future directions.

2. BACKGROUND

Image segmentation is the process of partitioning an image into multiple segments or objects aiming at analyzing and understanding the image and its context for computer vision recognition systems [15]. This section covers the essential background methods for image segmentation proposed in the literature and utilized in this paper.

2.1 Classical segmentation

Classical segmentation refers to methods that do not rely on deep learning algorithms. There is a broad spectrum of classical segmentation methods based on thresholding, morphological operators, edge detection, color space or background extraction. The segmentation algorithms, which are based thresholding techniques, measure the pixel intensity on the gray scale, then classify the scales based on global or local threshold values [16]. The segmentation methods based on edge detection detect discontinuity in the local features to generate maps with edges of objects [16]. On the other hand, the segmentation that relies on color spaces identifies different colors and maps them into separate classes.

Another approach for segmentation is based on background extraction algorithms. This approach can be used as a segmentation mechanism in applications where images are captured and provided as time series. In this technique, all detected objects that are static are classified as part of the background, while all moving particles are considered as foreground [17]. A Gaussian mixture-based (MOG2) model is proposed in [12, 13] utilizes properties of statistical distributions to create an improved background extraction algorithm that can overcome the complexity of the pixel value distribution in the image.

2.2 Deep learning segmentation models

Deep learning models have in recent years considerably increased performance for image segmentation tasks [15] over classical methods. The fully Convolutional network (FCN) proposed by Long *et al.* in [3] is considered a milestone for creating segmentation maps for images of varying dimensions using deep convolutional networks. However, the FCN is too slow when applied in real-time; moreover, it is not effective in capturing global contextual information [15]. In recent years, researchers have created a wide variety of deep learning models for image segmentation to improve the accuracy and efficiency of neural network models. This section evaluates a selection of deep learning models for semantic segmentation in the planktonic domain context.

2.2.1 U-net

U-net has obtained its name from the u-like structure of the network architecture. This architecture consists of one contracting path, an encoding path, and one expanding path, a decoding path. A key point in the U-net is that the feature maps generated in the contracting path are passed to the expanding path through skip connections. The skip connections help the expanding path construct the segmented output. The purpose of the contracting path is to capture contextual information, while the purpose of the expanding path is to construct the segmentation map output. The U-net architecture [4] was developed to solve a segmentation task with very little available training data, only 30 labeled images [4]. To generate a sufficient amount of training data, the author in [4] uses extensive data augmentation, and obtains significantly improved segmentation accuracy doing so.

2.2.2 Unet++

Unlike the Unet architecture, the U-net++ [6] introduces dense nested skip pathways between the encoder and decoder of the U-net architecture. The convolutional layers within the pathways are densely connected; this means that every neuron is connected to all neurons in the previous layer. The convolutional layers are preceded by concatenation layers which fuse the output from the preceding layer of the same dense block with the up-sampled output from the proceeding layer of the lower level dense block. This is done to make the encoder and decoder feature maps similar, under the assumption that this would lead to improved segmentation. U-net++ is in [6] trained and tested on four different datasets containing biomedical images. With the Intersection over Union (IoU) metric, it outperforms the original U-net on all four datasets.

2.2.3 Pyramid attention network

Just like the U-net, the Pyramid attention network [18] has an encoder decoder architecture. In addition, it utilizes a feature pyramid attention (FPA) module and global attention upsampling (GAU) modules. The proposed FPA module fuses features from different scales, to give more accurate segmentation. The GAU modules provides global context for decoding. Together with the encoder, these modules help the network capture global context, while at the same time they capture different scale of the feature information. A strength of the pyramid attention network is the ability to localize and classify small objects.

2.2.4 Deeplab V3+

The Deeplab V3+ uses dialations, so called atrous convolution, in the decoding of the segmentation maps [5, 19]. This enables, more computational efficient decoding, with a larger field of view for filters. In addition, Deeplab V3+ utilizes spatial pyramid pooling for robust segmentation of objects at multiple scales. Deeplab V3+ also combines methods from deep convolutional networks with probabilistic graphical models in order to improve the localization of object boundaries [5, 19].

2.2.5 Linknet

For real time applications that apply semantic segmentation, the run time is crucial. Linknet [20] is a simple network architecture designed to reduce the run time for predictions. It is 10 times faster than Segnet [21], another light-weight network that achieves higher accuracy for commonly used Cityscapes datasets [22].

3. RELATED WORK

Data augmentation is commonly used in computer vision to improve generalization for deep learning models. This applies especially when there are few labeled training images available [7]. A model with little available training data tends to overfit, meaning that the model is well adapted to the training set, while the performance drops significantly on the validation set. Data augmentation increases the number of data points for training, decreasing the distance between the training set and the validation set. This fact often yields improved model performance on the validation set [7]. There exists several methods to avoid overfitting for small datasets without data augmentation, such as batch normalization, dropout, pre-training, and transfer learning. These methods focus on the network architecture to increase the ability of generalization [7]. Data augmentation, on the other hand, handles the problem of small-sized training data at its core by creating additional training data.

Data augmentations is carried out through oversampling or data warping. Oversampling involves creating synthetic data through methods like feature space augmentations, mixing images, and generative adversarial networks (GANs) [7]. Data warping preserves the original labeling of the images. Geometric transformations, neural style transforms, random erasing, and color transformations are some examples of data warping [7].

3.1 Data augmentation using basic image manipulations

Two widely used groups of data warping manipulations are color space transformations and geometric transformations. The former transformations involve changing the color space within training images to make models more robust towards variations in lighting and color [23]. On the other hand, geometric transformations change geometric properties of the training images to make the models more robust to changes in position and orientation [23]. Examples of geometric transformations are rotation and flipping. Oversampling techniques using basic image manipulations include mixing images and random erasing. Mixing images combines sections of images into synthetic images [24]. Random erasing is done by selecting patches of training images while assigning all pixel values randomly or with predefined values [25]. Random erasing is done to overcome overfitting due to some objects or parts of images being unclear [7].

3.2 Data Augmentation using deep learning methods

Neural networks have the ability to map high dimensional inputs into lower dimensional representations [7]. Low dimensional feature maps can be extracted and isolated, opening up possibilities to use the neural networks in data augmentation [7]. Oversampling can be exerted by Generative adversarial network (GANs) as an example of deep neural network. GANs are used to generate artificial instances from a dataset while retaining similar characteristics of the original dataset. Another method is called neural style transfers can recreate an image so that it is displayed in a different style, while still retaining the original image motive [7].

4. METHODOLOGY

Proposed approaches for semantic segmentation having deep neural networks as their backbone architectures are extensively studied in the literature. This paper provides the experimental results based on the implementation, training, and performance comparison of the following network architectures.

- U-net [4]
- U-net++ [6]
- Linknet [20]
- Deeplab V3+ [5]
- Pyramid Attention Network (PAN) [18]

The training set consists of 284 images, and the validation set consists of 28 images. The labeling is done manually by domain experts, biologists. The original image size is 2448x2050. Figure 1 shows two input images along with their corresponding manually labeled masks. It is evident that the masks do not represent a perfect segmentation map. Figure 1 shows that the regions representing plankton in the masks are slightly bigger than the actual planktonic organisms sizes.

The different deep neural networks for segmentation, mentioned above, are trained over 15 epochs with batch size set to 4 using the dataset containing only manually labeled masks. The images and their masks are downsized to 512x512 as a preprocessing step before the training. The training is exerted over different loss functions in order to optimize the segmentation accuracy. Fine tuning the loss function is an important step in the training process since there is no ideal loss function that generalizes to all segmentation tasks[26]. The utilized loss functions in the process are listed as: binary cross entropy loss (BCE), weighted BCE, focal loss, and lovasz loss. For the U-net and the U-net++, the optimal loss function is the weighted BCE with positive weights set to 2. For the other network architectures, the optimal loss function is the weighted BCE with positive weight set to 7.

Each network is trained with its optimal loss function. Transfer learning is applied by importing resnet-101 [27] pre-trained on imagenet [28]. The learning rate is set to 0.0001 and the Adam optimizer is used in the algorithm [29]. The following standard data augmentation is used for all images: rotation (0°-35°), horizontal flip and vertical flip. We compare dice coefficient, precision and recall for all networks. The results from this comparison is used to decide which network to use for further testing with a larger dataset containing masks generated by Gaussian mixture-based (MOG2) segmentation.

We modify two selected networks by using different encoders. This is done to see if we can obtain less complex models giving faster predictions without significantly decreasing the dice coefficient. In real time image processing, run time is

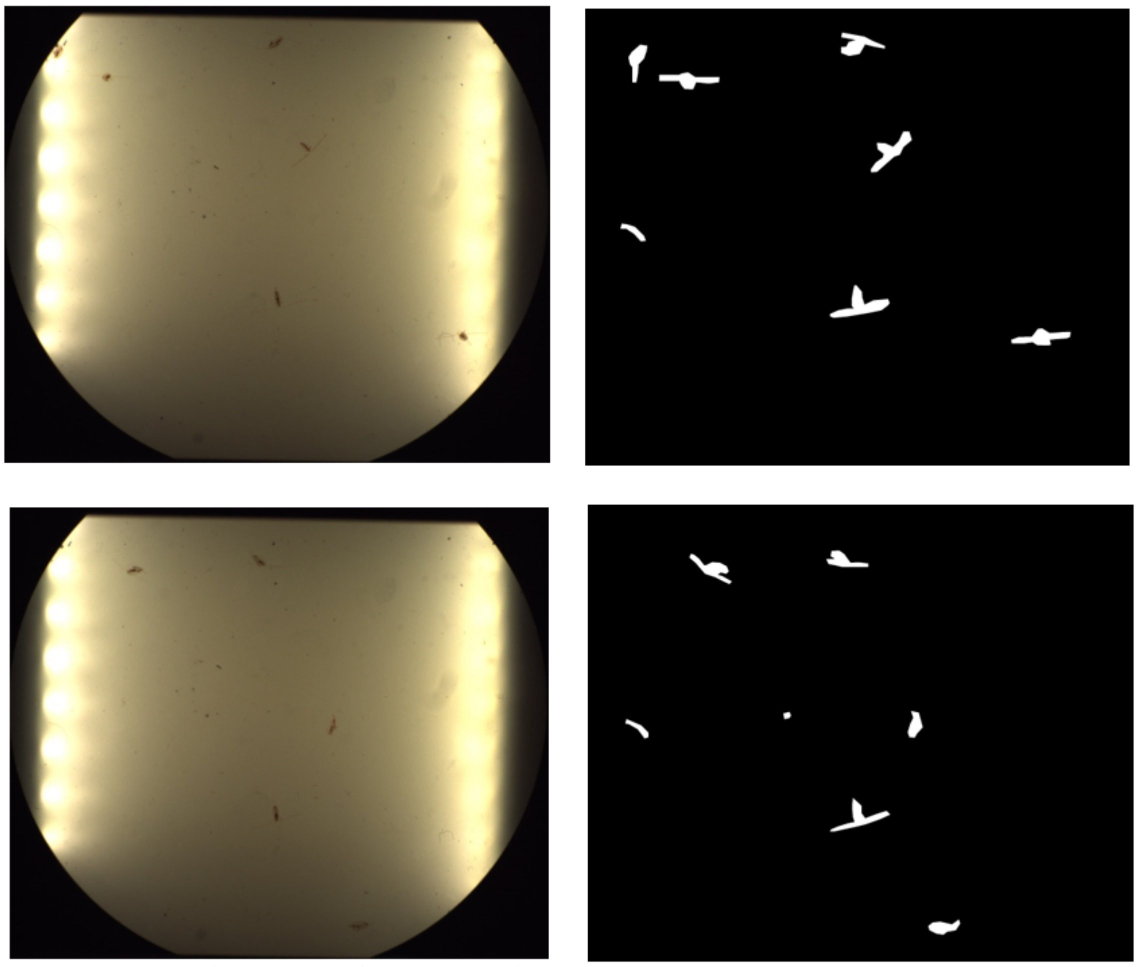


Figure 1: Input images and their corresponding manually labeled masks.

an important factor. We found that when using the pre-trained resnet 101 encoder, the Unet++ yields the highest dice coefficent, while the Linknet yields lowest run-time for prediction of images. We do further training for the U-net++ and the Linknet using different encoders. For U-net++ and Linknet we employ the following encoders; resnet 101 [27], resnet 34 [27], and mobilenet V2 [30].

The labeled dataset utilized is rather small with only 312 labeled images. Therefore we explore the use of Gaussian mixture-based (MOG2) segmentation to generate more training data. We carry out dilation [14] and erosion [14] to remove noise from the output masks. We use a 10×10 kernel for dilation and a 20×20 kernel for erosion. Gaussian mixture-based (MOG2) segmentation followed by dilation and erosion is carried out on the training data set, and can in that regard be viewed as a special case of data augmentation. Figure 2 shows the output from MOG2 before and after the removal of noise. We observe that 19% of the generated masks represent poor foreground and background segmentation; an example from the poor quality generated masks is shown in figure 3.

5. EXPERIMENTAL RESULTS

As a result of the training process, we observe that the Unet++ achieves the highest overall performance metrics when we train all networks using only manually labeled masks. We therefore choose the U-net++ for further training using the larger dataset containing both types of masks manually labelled, and masks generated through Gaussian mixture-based segmentation (MOG2). The hyper-parameters were fixed and the pre-trained resnet 101 is used as the encoder architecture. We measure the network accuracy using the dice coefficient, precision and recall.

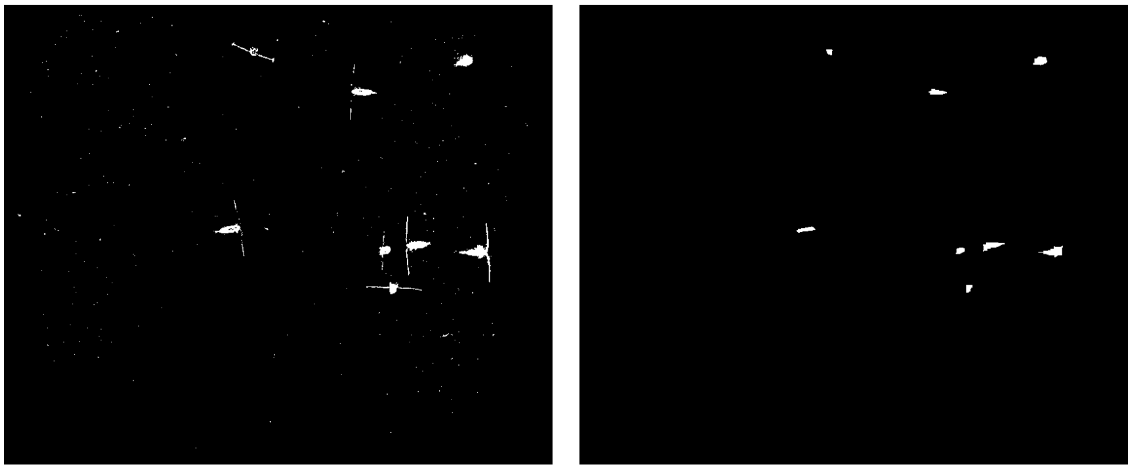


Figure 2: MOG2 output before and after noise removal

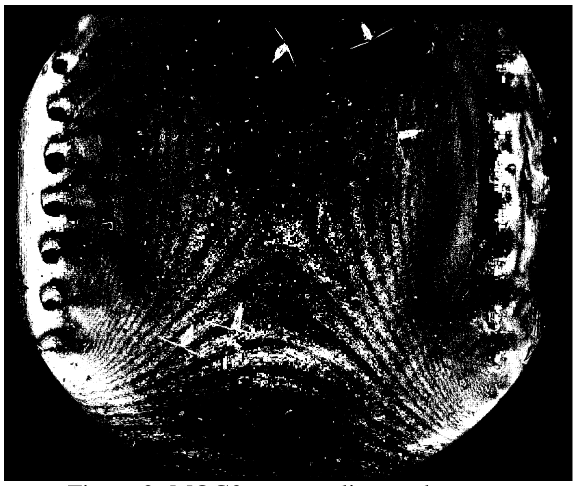


Figure 3: MOG2 poor quality mask output

We observe that the dice score decreases when we use the larger training set generated by the MOG2 algorithm. However we have seen that our original training data is not perfectly labeled. The masks have a slight overweight of false positives. The performance metrics, dice, precision and recall, are calculated by comparing the output masks from the model with manually labeled masks. This implies that a perfect segmentation will have high precision score, and a lower recall score. We observe that with our proposed MOG2 data augmentation we obtain increased precision and lowered recall. Figure 4 shows that this leads to improved segmentation. It is worth noting, that the antennas of the planktonic organisms in the masks to the right are thinner than those presented in the mask to the left. In reality, antennas of this species of plankton, copepods, are very thin. Both masks show accurate segmentation of the plankton species bodies.

Network architecture	Dice	Precision	Recall
U-net	0.7755	0.7704	0.7821
U-net++	0.8142	0.7837	0.8493
Linknet	0.7720	0.6756	0.9030
Deeplab V3+	0.7441	0.6558	0.8610
PAN	0.7515	0.6631	0.8696

Table 1: Performance metrics of the networks trained on the manually labeled dataset

Network architecture	Dice	Precision	Recall
U-net++	0.7779	0.8527	0.7170

Table 2: Performance metrics of the networks trained on the large dataset

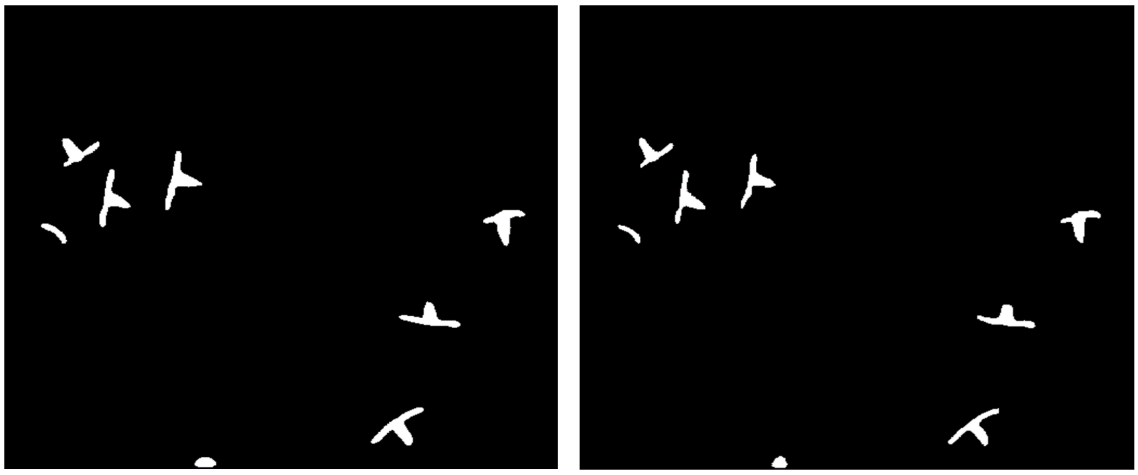


Figure 4: Resulting output masks U-net++. To the left is the result without applying the MOG2 augmentation, to the right the result after applying the MOG2 augmentation

6. CONCLUSION

This paper uses Gaussian mixture-based (MOG2) segmentation as a data augmentation approach to generate additional labeling in the training dataset used by deep learning attention segmentation models in their training process. The method is specifically applied to applications which provide captured images in a time series format. The idea is to extract the foreground from the background through excluding objects that are statically presented or not changing their positions in images placed in a sequence. As an example, we showcase the applicability of the method to the planktonic domain and more specifically to the platform described in [8]. We implement and train several deep neural network architectures for segmentation, and we train the networks using different loss functions to optimize their accuracy. Experimental results show that the use of the manually labelled masks augmented by the generated masks from the MOG2 yields improved segmentation. An interesting future direction would be to investigate other background extraction algorithms and use them for data augmentation. Another interesting future direction would be exploring the effect of this type of data augmentation on the different network architectures to determine which combination is more suited to which specific context and application.

ACKNOWLEDGMENTS

This research is funded by the Research Council of Norway (RCN) IKTPLUSS program (project number 262741) and supported by NTNU AMOS (RCN project number 223254).

REFERENCES

[1] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, “Semantic segmentation using regions and parts,” in 2012 IEEE conference on computer vision and pattern recognition, 3378–3385, IEEE (2012).

[2] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang, “Attention-guided unified network for panoptic segmentation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7026–7035 (2019).

[3] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 3431–3440 (2015).

[4] 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, 234–241, Springer (2015).

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in Proceedings of the European conference on computer vision (ECCV), 801–818 (2018).

[6] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in Deep learning in medical image analysis and multimodal learning for clinical decision support, 3–11, Springer (2018).

[7] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," Journal of Big Data **6**(1), 1–48 (2019).

[8] A. Saad, A. Stahl, A. Våge, E. Davies, T. Nordam, N. Aberle, M. Ludvigsen, G. Johnsen, J. Sousa, and K. Rajan, "Advancing ocean observation with an ai-driven mobile robotic explorer," Oceanography **33**(3), 50–59 (2020).

[9] S. Bergum, A. Saad, and A. Stahl, "Automatic in-situ instance and semantic segmentation of planktonic organisms using mask r-cnn," in Global Oceans 2020: Singapore–US Gulf Coast, 1–8, IEEE (2020).

[10] A. Saad, S. Bergrum, and A. Stahl, "An instance segmentation framework for in-situ plankton taxa assessment," in Thirteenth International Conference on Machine Vision, **11605**, 1160511, International Society for Optics and Photonics (2021).

[11] P. Falkowski, "Ocean science: the power of plankton," Nature **483**(7387), S17–S20 (2012).

[12] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., **2**, 28–31, IEEE (2004).

[13] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," Pattern recognition letters **27**(7), 773–780 (2006).

[14] G. Bradski and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library, " O'Reilly Media, Inc." (2008).

[15] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).

[16] S. Yuheng and Y. Hao, "Image segmentation algorithms overview," arXiv preprint arXiv:1707.02051 (2017).

[17] C. Wren, "Real-time tracking of the human body," SPIE proceeding, 1996 **2615**, 89–98 (1996).

[18] H. Li, P. Xiong, J. An, and L. Wang, "Pyramid attention network for semantic segmentation," arXiv preprint arXiv:1805.10180 (2018).

[19] 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 and machine intelligence **40**(4), 834–848 (2017).

[20] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," in 2017 IEEE Visual Communications and Image Processing (VCIP), 1–4, IEEE (2017).

[21] 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 **39**(12), 2481–2495 (2017).

[22] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in Proceedings of the IEEE conference on computer vision and pattern recognition, 3213–3223 (2016).

[23] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," arXiv preprint arXiv:1708.06020 (2017).

[24] H. Inoue, "Data augmentation by pairing samples for images classification," arXiv preprint arXiv:1801.02929 (2018).

[25] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in Proceedings of the AAAI Conference on Artificial Intelligence, **34**(07), 13001–13008 (2020).

[26] S. Jadon, "A survey of loss functions for semantic segmentation," in 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 1–7, IEEE (2020).

[27] 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, 770–778 (2016).

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition, 248–255, Ieee (2009).

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

[30] 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, 4510–4520 (2018).