Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Lars Giske

Automated Video Segmentation in Underwater Ship Inspections Exploiting Deep Learning, Computer Vision, and Image Enhancement

Master's thesis in Marine Technology
Supervisor: Martin Ludvigsen
Co-supervisor: Asgeir J. Sørensen and Alexandre Cardaillac
June 2022

Lars Giske

Automated Video Segmentation in Underwater Ship Inspections Exploiting Deep Learning, Computer Vision, and Image Enhancement

Master's thesis in Marine Technology
Supervisor: Martin Ludvigsen
Co-supervisor: Asgeir J. Sørensen and Alexandre Cardaillac
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER OF TECHNOLOGY THESIS DEFINITION (30 SP)

Name of the candidate:	Lars Giske
Field of study:	Marine Cybernetics
Thesis title (Norwegian):	Automatisert videosegmentering for undervanns skipsinspeksjoner ved bruk av dyplæring, datasyn og bildeforbedring
Thesis title (English):	Automated video segmentation in underwater ship inspections exploiting deep learning, computer vision, and image enhancement

Background

Underwater ship inspections, which have in general been performed by professional divers, provide a cheaper and less time-consuming alternative to inspections through dry docking. In recent years, the use of ROVs (remotely operated vehicles) has been used instead of divers, which will remove the risk of accidents. The ROVs need an expert operator and manual navigation, which can be expensive and time-consuming. Later, autonomous underwater vehicles (AUVs) have been an increasingly popular technology, since they can remove the chances for human errors and intervention. The AUVs and ROVs are usually equipped with cameras to perform the inspection. The videos from the inspection normally must be analyzed manually. Live automatic processing of the video can be beneficial in several aspects. It can be used to create an objective and data-driven evaluation of the underside of the ship and find areas of interest that must be inspected further, which can assist the inspector during evaluation. A convolutional neural network can be trained on a relevant image set to be able to do semantic segmentation and classification of the images of the ship to find areas to locate areas of interest with corrosion, paint peel, marine growth, and defects. Underwater ship inspections have become more important in the last years, because of the implementation of stricter regulations. Marine growth will create more drag and increase fuel consumption and paint peel can increase the marine growth. One aspect of underwater data-driven segmentation which can cause poor

Scope of Work

1. Perform a background study to provide information and relevant references on:
    - Machine learning with a focus on semantic segmentation based on visual data.
    - Improvement and enhancement of underwater imagery.
    - Architectures and structures of convolutional neural networks.
    - Relevant computer vision methods for video processing
2. Compare different deep learning-based semantic segmentation models and their prediction performance on the LIACI dataset.
3. Find the convolutional neural network for semantic segmentation best suited for video segmentation in an underwater ship scenario based on the comparisons in 2.
4. Consider if it is possible to improve the CNN by altering or augmenting the LIACI dataset.
5. Propose a method for utilizing the shared information between two frames in a video to improve the segmentation model from 3, either by additional training of the model or by a computer vision-based method.
6. Implement methods for underwater image enhancement and investigate if the enhancement methods can increase the performance of the segmentation model in 3 and the method in 4
7. Combine 3 with the methods from 4,5 and 6 and test the complete model on provided videos of underwater ship inspection.

Specifications

The student shall at startup provide a maximum 2-page week plan of work for the entire project period, with main activities and milestones. This should be updated on a monthly basis in agreement with supervisor.

Every weekend throughout the project period, the candidate shall send a status email to the supervisor and co-advisors, providing two brief bulleted lists: 1) work done recent week, and 2) work planned to be done next week.

The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences about grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, problem/research statement, design/method, analysis, and results. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts. It shall be written in English (preferably US) and contain the elements: Title page, abstract, preface (incl. description of help, resources, and internal and external factors that have affected the project process), acknowledgement, project definition, list of symbols and acronyms, table of contents, introduction (project background/motivation, objectives, scope and delimitations, and contributions), technical background and literature review, problem formulation, method, results and analysis, conclusions with recommendations for further work, references, and optional appendices. Figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation or Vancouver reference style (e.g. natbib Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct, which is taken very seriously by the university and will result in consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed.

The thesis shall be submitted with an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis definition shall be included after the title page. Computer code, pictures, videos, data series, etc., shall be included electronically with the report.

Start date:	15.01.2022
Due date:	11.06.2022

Supervisor:	Martin Ludvigsen
Co-advisor(s):	Asgeir J. Sørensen, Alexandre Cardaillac

Trondheim, 10.06.2022

[illegible]

Professor Martin Ludvigsen
Supervisor

Abstract

Ship hull inspections are a crucial part of managing and maintaining the standard of aging ships. In-water ship inspections, performed by divers or remotely operated vehicles (ROVs), can be a viable alternative to dry dock inspections. A dry dock inspection is far more expensive and time-consuming than an underwater ship inspection. Therefore, with the progress of unmanned underwater vehicles, underwater ship inspections have received more attention in recent years. Ship inspections are done visually to discover defects and get an overview of the state of the ship, by evaluating the amount of growth, paint peel, and corrosion present. The underwater inspections are done manually by visually evaluating a video taken by the ROV. Automating this task can reduce the complexity, cost, and inspection time, which is the main motivation for the work of this thesis.

The thesis focuses on the use of deep learning in combination with image enhancement and computer vision to perform pixel-wise semantic segmentation of a ship hull during underwater ship inspections. The training is done on the LIACI dataset, which contains 1893 images with ten corresponding labels representing objects of interest commonly encountered during an underwater ship inspection.

This thesis presents a method for increased performance and consistency during an automated ship inspection, consisting of a semantic segmentation model based on deep supervised learning, image enhancing, and prediction averaging based on computer vision. A deep learning model is proposed based on comparisons with state-of-the-art models and other versions of the proposed model. The proposed model achieves an IoU improvement of 3.8% compared to the standard model in a five-fold cross-validation test. A method is proposed for better video segmentation utilizing the information from the previous prediction by using image matching, 2D transformation, and an adaptive weighted average between the previous and current prediction. This method showed more concise and accurate predictions than evaluating each frame separately. Image enhancement is applied to test if it can increase the performance of a segmentation model and increase the number of well-matched features between two images. Seven image enhancement techniques were tested and compared, where two methods increased the performance of both the segmentation model and the model for 2D transformations. A generative adversarial network was trained to augment the training set with generated images based on segmentation masks. Training on the augmented dataset showed promising results with few real images in the training set, but poorer results when augmenting the dataset based on masks already present in the dataset.

Videos of underwater ship inspections were used to test the capability of video segmentation during a realistic underwater inspection scenario. The final proposed method, combining machine learning, image enhancement, and computer vision, showed significantly better performance than the standard machine learning method. The video segmentation done by the final proposed model was able to perform at a satisfactory level, showing the possibilities and the potential of automated underwater ship inspections. The computer vision method had a large positive impact on the total performance during video segmentation and proves that utilizing additional position information is important for precise segmentation during visual inspections.

Sammendrag

Inspeksjon av skipsskrog er en avgjørende del av å administrere og opprettholde standarden på aldrende skip. Undervannsinspeksjoner av skip, utført av dykkere eller fjernstyrte undervannsdroner (ROVs), kan være et godt alternativ til tørrdokkinspeksjoner. En tørrdokkinspeksjon er langt dyrere og mer tidkrevende enn å uføre en undervannsinspeksjon. På grunn av fremgangen til ubemannede undervannsfarkoster, har undervannsinspeksjoner fått mer oppmerksomhet de siste årene. Skipsinspeksjoner gjøres visuelt for å oppdage defekter og få oversikt over skipets tilstand. Dette kan gjøres ved å evaluere mengden begroing, malingsavskalling og korrosjon på skroget og andre deler av skipet. Undervannsinspeksjonene gjøres manuelt ved å visuelt evaluere video og bilder som blir samlet av en ROV. Automatisering av dette arbeidet kan redusere kompleksiteten, kostnadene og inspeksjonstiden, som er hovedmotivasjonen for arbeidet med denne oppgaven.

Oppgaven fokuserer på bruk av dyplæring kombinert med bildeforbedring og datasyn for å utføre pikselvis semantisk segmentering av et skipsskrog under en undervannsinspeksjon. Treningen av dyplæringsmodellen gjøres på LIACI-datasettet, som inneholder 1893 bilder med ti tilsvarende klasser som representerer gjenstander som ofte støtes på under en undervannsinspeksjon.

Denne oppgaven presenterer en metode for økt ytelse og kontinuitet under en automatisert skipsinspeksjon, bestående av en semantisk segmenteringsmodell basert på dyp overvåket læring, bildeforbedring, og prediksjonsgjennomsnitt basert på datasyn. En dyplæringsmodell foreslås basert på sammenligninger med moderne modeller og andre versjoner av den foreslåtte modellen. Den foreslåtte modellen oppnår en IoU-forbedring på 3,8% sammenlignet med standardmodellen i en femdobbel kryssvalideringstest. En metode er foreslått for bedre videosegmentering ved å utnytte informasjonen fra forrige prediksjon ved å bruke bildematching, 2D-transformasjon, og et adaptivt vektet gjennomsnitt mellom forrige og nåværende prediksjon. Denne metoden viste mer konsise og nøyaktige prediksjoner enn å evaluere hver ramme separat. Bildeforbedring brukes for å teste om det kan øke ytelsen til en segmenteringsmodell og øke antallet gode matcher mellom to bilder. Syv bildeforbedringsteknikker ble testet og sammenlignet, der to metoder økte ytelsen til både segmenteringsmodellen og modellen for 2D-transformasjoner. Et generativt konkurrerende nettverk (generative adversarial network) ble opplært til å utvide treningssettet med genererte bilder basert på segmenteringsmasker. Trening på det utvidede datasettet viste lovende resultater med få reelle bilder i treningssettet, men dårligere resultater ved utvidelse av datasettet basert på masker som allerede finnes i datasettet.

Videoer av undervannsinspeksjoner av skip ble brukt til å teste evnen til videosegmentering under et realistisk undervannsinspeksjonsscenario. Den endelige foreslåtte metoden, som kombinerer maskinlæring, bildeforbedring og datasyn, viste betydelig bedre ytelse enn standard maskinlæringsmetode. Videosegmenteringen gjort av den endelige foreslåtte modellen var i stand til å yte på et tilfredsstillende nivå, og viste mulighetene og potensialet for automatiserte undervannsskipsinspeksjoner. Datasynsmetoden hadde en stor positiv innvirkning på den totale ytelsen under videosegmentering, og beviser at bruk av ekstra posisjonsinformasjon er viktig for presis segmentering under visuelle inspeksjoner.

Preface

The research presented in this master's thesis is the result of independent work done by the author during the spring of 2022. This thesis concludes my Master's degree in Marine Technology at the Norwegian University of Science and Technology (NTNU), with a specialization in Marine Cybernetics. It represents work done in TMR4930 - Marine Technology, Master's Thesis, accounting for 30 ETC.

The work is inspired by my project thesis in the fall of 2021, where the field of underwater predictions with machine learning was one of my focus areas. During the preparations for the master's thesis, my supervisor Professor Martin Ludvigsen suggested basing the thesis on a newly presented dataset for underwater ship inspection. Professor Martin Ludvigsen and co-supervisor Alexandre Cardaillac provided access to the dataset and other relevant videos, which were used as the basis for this thesis.

Acknowledgements

I would like to thank my supervisor, Professor Martin Ludvigsen for is excellent assistance, guidance, and advice throughout the work of this thesis. I would also like to thank my co-supervisor Alexandre Cardaillac. He has also provided with a deeper understanding and expert knowledge in the material I have been working with. Both Professor Martin Ludvigsen and Alexandre Cardaillac has given me good, insightful help with challenges I have had and suggested well-needed ideas for interesting and promising topics areas to further look into during this period. Additionally, I would like to express my gratitude to my co-supervisor, Professor Asgeir Sørensen for is motivating help in the meetings during the project thesis during the fall 2021, leading up to this master's thesis.

The work in this thesis has mainly been conducted on Tyholt at NTNU. Here, I have shared an office with four fellow master students. The office have been a productive place to work, and helped with motivation in the last year. Therefore, I would like to thank the persons from this office, C1.084, for advice when needed and also motivation-increasing ping-pong matches, especially during the long days towards the end of this period.

Trondheim, June 10, 2022
Lars Giske

Table of Contents

List of Tables

List of Figures

Abbreviations

ANN	=	Artificial neural network
AHE	=	Adaptive histogram equalization
API	=	Application programming interface
BRIEF	=	Binary Robust Independent Elementary Features
BRISK	=	Binary robust invariant scalable keypoints
CII	=	Carbon intensity indicator
CNN	=	Convolutional neural network
cGAN	=	Conditional generative adversarial network
CPU	=	Central Processing Unit
CLAHE	=	Contrast limited adaptive histogram equalization
DCP	=	Dark channel prior
EUVP	=	Enhancing Underwater Visual Perception
FCN	=	Fully convolutional network
FLANN	=	Fast library for approximate nearest neighbors
FPS	=	Frames per second
GAN	=	Generative adversarial networks
GPU	=	Graphics processing unit
HSE	=	Hue, saturation, intensity
ICM	=	Integrated color model
IoU	=	Intersection over union
LIACI	=	Lifecycle inspection, analysis and condition information
LSTM	=	Long short-term memory
ORB	=	Oriented FAST and rotated BRIEF
RANSAC	=	Random sample and consensus
ReLU	=	Rectified linear unit
RGB	=	Red, green blue
RCNN	=	Recurrent convolutional neural network
RNN	=	Recurrent neural network
ROV	=	Remotely operated vehicle
SIFT	=	Scale invariant feature transform
SLAM	=	Simultaneous localization and mapping
SOTA	=	State-of-the-art
SUIM	=	Semantic segmentation of underwater imagery
SURF	=	Speeded-up robust features
TLFN	=	Two-hidden-layer feedforward network
UWILD	=	underwater inspection in lieu of dry-docking

Chapter 1

Introduction

1.1 Background and Motivation

Underwater operations have been an important part of the offshore oil and gas industry, starting in the 1960s. In the first twenty-five years of the growing underwater industry, humans divers performed these operations. They were dangerous and time-consuming for the divers. Approximately 375 of the divers in the North Sea in this time period were Norwegian, and there were 17 accidents with a fatal outcome. In a study from 2004 in including 230 Norwegian divers, 96 of the participants still had work related injuries Fanebust and Smith-Solbakken (2021). In the later years, unmanned vehicles have been used to perform intervention and survey tasks. With unmanned underwater vehicles, the same tasks can be performed safe, and more cost-effective. Other tasks can also be performed more efficiently and safer with unmanned underwater vehicles, such as mapping and surveying.

The interest in underwater ship inspections have increased over the past years. The advantage of underwater ship inspection is that it is cheaper and less time-consuming than the alternative, which is inspection through dry docking. Another important factor for increased interest, is for better fuel efficiency. Marine growth can lead to an increased fuel consumption of 10-20% (Hakim et al., 2018). This will not only lead to a 10-20% increase in fuel price, but can also restrict a ship from sailing with new regulations from 2023, when the Carbon Intensity Indicator (CII) will take effect for all cargo, RoPax and cruise vessels above 5,000 gross tonnage and trading internationally (DNV, 2022b). According to Ship Review (Scope Group, 2021) and 2020 data, 21.8% of all ships affected by the CII requirements needs to improve their rating to be allowed to sail in 2025. Performing ship inspections can identity if other actions are needed to be taken in regard to safety by locating defected areas, or to increase the fuel efficiency by locating with marine growth, paint peel and corrosion. The inspection is normally done manually by visually evaluating the video from the ROV that is used for inspection. Manual evaluation is time-consuming, and it can be difficult to quantify the observations precisely. This process can be automized by using a recent progress withing the field of deep learning for image processing. The process of providing a label to each pixel in a picture is known as semantic segmentation. In a ship inspection, semantic segmentation can be used for locating objects of interest, such as marine growth, paint peel corrosion and defects. Other objects can also be localized, such as propellers, anodes, valves, sea chest gratings and keels to get data on where marine growth, paint peel corrosion and defects occurs. Using semantic segmentation to automate the visual survey and inspection can be used as a decision support system by producing measurable, objective and detailed data during the inspection.

1.1.1 Problem Formulation

Underwater ship inspections is a challenging field commonly performed manually by an expert surveyor, which is time-consuming and requires subjective analysis, where poor visibility can be an issue. An automatic approach can act as decision support for the surveyor by providing quantitative objective measures with more possibilities for post-processing and real-time data-driven analysis of the data. The main focus of this thesis will be to formulate a method for automatic visual ship inspection for live localization on pixel level and classification of relevant objects and elements.

1.2 Research Question and Objectives

This thesis examines the combination of deep learning based methods, computer vision and image enhancement to obtain information about the state of a ship during underwater ship inspection based on visual data from an ROV camera. The data available for this research is a large-scale dataset for underwater ship lifecycle inspection and videos from underwater ship inspections. Hence, the purpose of this thesis can be summarized with the following research questions.

1. What deep learning models are suitable for real-time segmentation to obtain information about the state of a ship hull at pixel-level?
2. How can the correlation between frames in a video be utilized to improve the obtained information extracted from the video?
3. Can image enhancement methods be applied to increase the quality of predictions and estimations?
4. Can the performance of class predictions be increased by augmenting the existing dataset?
5. How can the methods be combined to obtain an overall increased performance in automatic ship inspection analysis?

These research questions will be examined by the following objectives:

1. Perform a literature study and provide theory on existing methods in the topics:
    - Deep neural networks.
    - Semantic segmentation based on deep supervised learning.
    - Image enhancement with focus on the underwater scene.
    - Computer vision with focus on 2D transformations and feature matching.
2. Implement existing deep learning based models for semantic segmentation, with and without alternations.
3. Train the implemented models on the LIACI dataset for performance comparison.
4. Implement existing enhancement methods and train a model with enhanced images to investigate performance improvements.
5. Implement an existing conditional generative adversarial network and customize the model for generating images based on the annotated masks in the LIACI dataset.

6. Train the adversarial model on the LIACI dataset to investigate if augmenting the dataset with generated images can increase the prediction performance of a segmentation model.
7. Implement computer vision method to utilize previous predictions during video segmentation.
8. Investigate if enhanced images will increase accuracy of the computer vision method.
9. Analyze the results of the methods tested and combine the chosen methods for to optimize the performance of semantic segmentation predictions on underwater ship inspection videos.
10. Suggest recommendations of further work based on results in this thesis.

1.3 Related Work

This section contains a brief introduction of related work regarding in-water ship inspections.

Underwater Inspection in Lieu of Dry-Docking (UWILD) is an alternative to conventional dry docking for inspection of marine vessels. UWILD can reduce downtime, cost and additional travel time. UWILD is conventionally performed by divers, but performing inspection with ROVs have become a viable alternative (Antony Jacob, 2018). Antony Jacob (2018) presents a UWILD survey where damages, marine growth on the keel and gratings and irregular pant conditions on the rudder and hull were noticed. These observations were made live by a class surveyor and representatives from the vessel management company, and action plans were made based on the observations.

Progress towards autonomous navigation and planning of UWILD inspections has been made. In Hover et al. (2012) algorithms are developed for navigation and planning problems in UWILD inspection, using visual and acoustic mapping processes, for full image coverage of the structure.

Waszak et al. (In Press) presents the first large-scale dataset for semantic segmentation in underwater ship inspection and proposes a segmentation model for this use-case. The dataset is the basis of this thesis. No other work regarding semantic segmentation in a UWILD scenario in known to the author. In (Islam et al., 2020a), semantic segmentation is used in an underwater scene, where a relevant model to this thesis is proposed. O'Byrne et al. (2018) proposes the use of generated imagery for training deep models to interpret real-world underwater imagery, where the problem of marine growth detection on marine structures is looked at.

Corrosion is important to localize and evaluate during a UWILD inspection and in inspections of structures in general. DNV offers a tool for corrosion inspection and monitoring, called corrosion.ai, with the use of semantic segmentation to replace time-consuming and costly close-up visual inspection (DNV, 2022a). Corrosion detection through semantic segmentation is also looked at in Petricca et al. (2016), where a deep learning model outperformed a classic computer vision technique in corrosion detection.

1.4 Scope and Delimitations

In this thesis, several delimitations have been made to narrow the scope of the thesis. The main delimitations are:

- All the segmentation models tested are based on the VGG16 feature extractor as a backbone. Other feature extracting convolutional neural nets mentioned in this thesis are not implemented or tested.

One model will be chosen as the baseline and the focus have been to improve performance based on this.

- 3D transformations are not considered, and the number of good matches are the only considered metric for estimating the accuracy of the homography. Evaluation methods such as projection error are not considered.
- Live analysis is the main application for this thesis, but the implemented code method is not optimized for efficiency.

1.5 Contributions

To summarize this work, the contributions can be listed as follows:

- A proposed version of the existing segmentation model SUIM-Net$_{vgg16}$, with added attention gate in the deepest layer, which achieved the best performance in the results.
- Combination of image enhancement and semantic segmentation training, which showed promising results for two model-free methods.
- Results showing the possibilities and current limitations of utilizing a GAN for augmenting a dataset for better segmentation.
- Proposed method for increasing the prediction performance of a segmentation model when a video is used as input, utilizing overlapping frames. The model is based a weighted average between the predictions at the previous frame and the current prediction using 2D transformation. The weighting is decided based on the reliability of the estimated transformation matrix between two consecutive frames. This method is not directly based on existing methods for this application.
- Utilizing the image enhancement methods for increasing the reliability of the estimated transformation matrix, where the same two model-free methods achieved significant performance improvement.
- A combined model for video segmentation, where image enhancement is used for segmentation and feature matching, the proposed segmentation model is used for predictions and the proposed computer vision method is used for improvement of the estimates from the segmentation model.
- Suggestions for further work.

1.6 Outline

The following chapters will be structured as follows:

Chapter 2 introduces, describes and studies relevant theory, methods, and literature for this thesis. The topics covered are neural networks with a focus on convolutional neural networks for semantic segmentation, image enhancement with a focus on the underwater scene, and computer vision with a focus on 2D image matching.

Chapter 3 describes the methods used in this project and what data the methods are trained, evaluated and tested with. The implementations are described, including data processing, deep-learning models, image enhancement methods, the training setup for the supervised and unsupervised learning, and the algorithm for image matching used for weighted average predictions. The chapter will also inform about existing frameworks and code used in the implementations.

Chapter 4 presents the results from this thesis and the discussion and analysis of the results. The results contain comparisons and performance tests based on quantitative measurements and visual examples.

Chapter 5 discusses inconsistencies and other aspects that can have an influence on the results and provides a further discussion on the results.

Chapter 6 concludes the findings from the results and proposes suggestions for further work.

Chapter 2

Literature Study and Theory

2.1 Deep Neural Networks

2.1.1 Artificial Networks

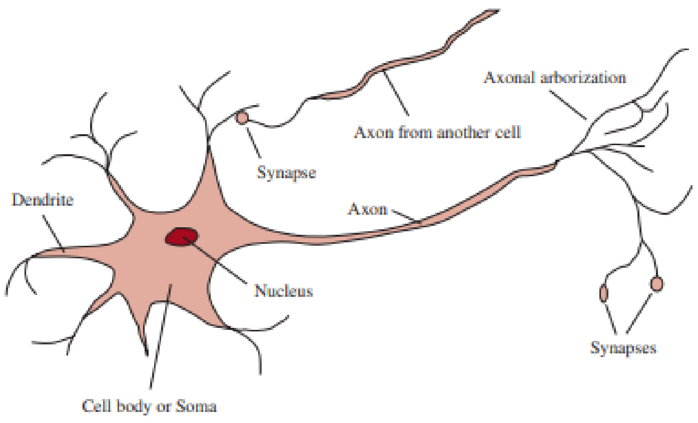


Figure 2.1: Neuron. Courtesy: Russell and Norvig (2019)

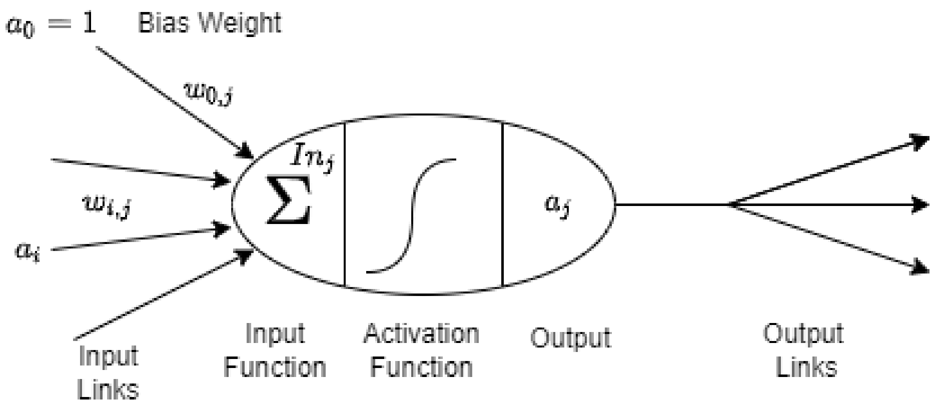


Figure 2.2: Mathematical model of a neuron

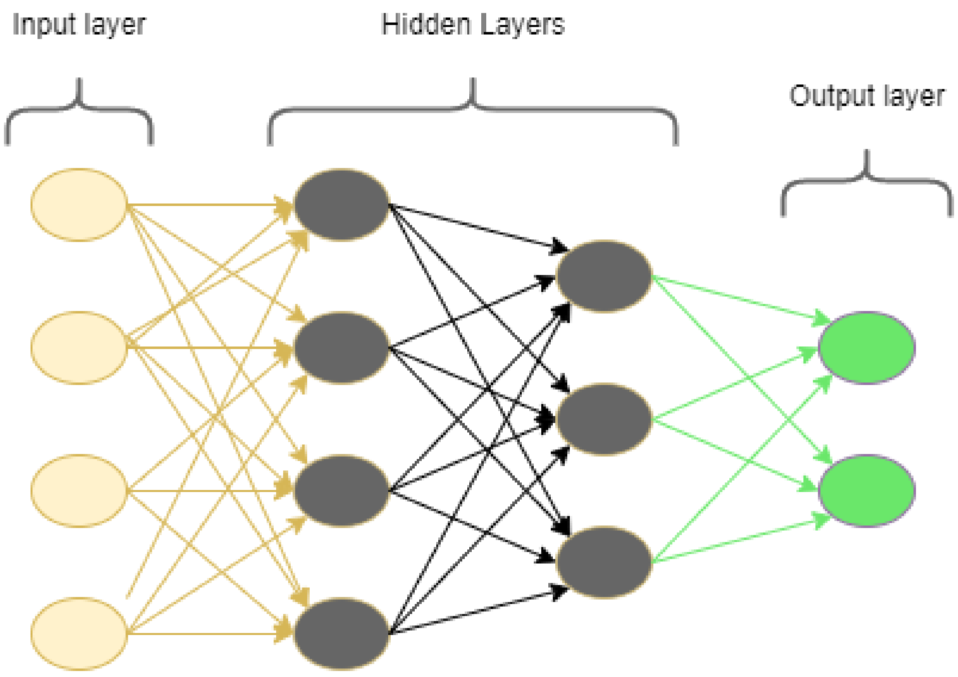


Figure 2.3: Neural network

Artificial neural networks (ANNs) have been a popular topic within the field of artificial intelligence since the 1980s. An ANN consists of connected neurons (or nodes) as an abstraction of the human brain. Figure 2.1 shows a simple representation of a nerve cell, or neuron. The Axon is a single long fiber branching out from the cell body. The axon is connected to 10 to 100,000 other neurons through junctions from the Axon. These junctions are called synapses. Signals travel between the neurons by electrochemical reactions. The electrochemical reactions control the connectivity of the neurons and form the basis of learning (Russell and Norvig, 2019). The first mathematical model of a neuron was devised by McCulloch and Pitts in 1943 (Russell and Norvig, 2019) and more advanced and improved versions have been proposed since then. Artificial neurons can be combined to through connections to form ANNS. ANNs have a broad application area and have been used across different domains, such as object detection, classification, speech recognition, segmentation, detection, and prediction. Figure 2.2 shows a mathematical model of a neuron and is the basis in artificial neural networks.The input function is the sum of the weighted inputs:

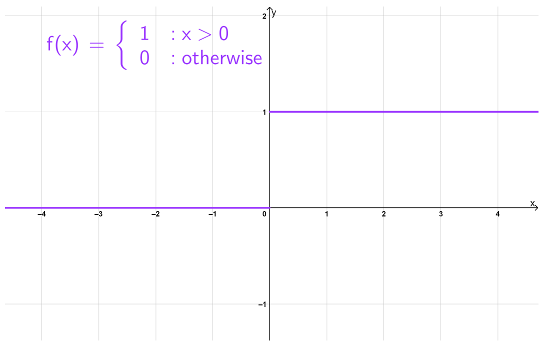
$$Z = \sum_{i=0}^{n} w_{i,j} a_i \tag{2.1}$$

where a_i is the output activation of node *i* and $w_{i,j}$ is the weight from unit i to this node j. The node's output activation function is:

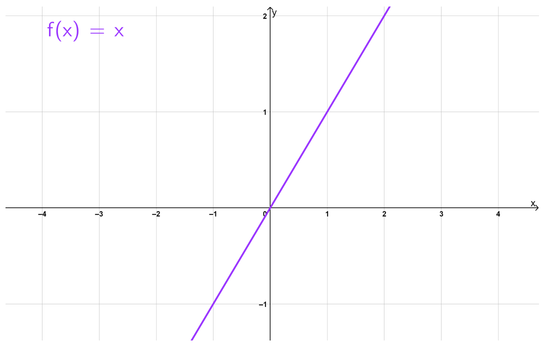
$$a_j = g(Z) \tag{2.2}$$

The activation function can either be linear or non-linear and used to control the outputs of the neural network. Non-linear activation functions are needed to convert the linear input to non-linear output, which makes it possible for the network to learn high order polynomials beyond one degree of freedom (Chigozie Nwankpa and Marshall, 2018). Four examples for activation functions are shown in Figure 2.4, where ReLU, sigmoid, and softmax are common activation functions.

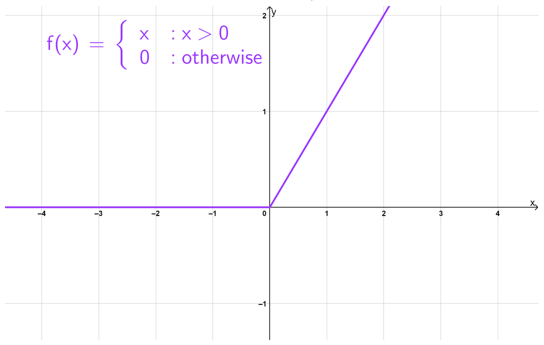
Activation Functions



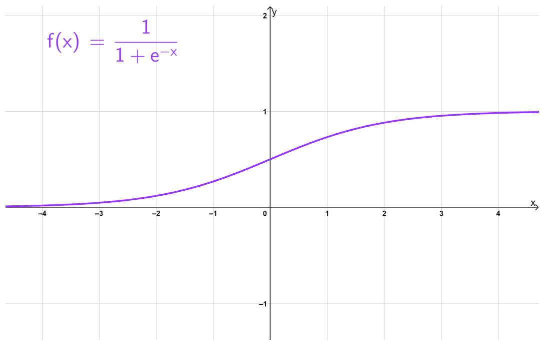
(a) Binary



(b) Linear



(c) ReLU



(d) Sigmoid

Figure 2.4: Activation functions

- **Binary:** Function shown in Figure 2.4a. This is the simplest activation function and can be used in binary classifiers. The binary activation function can not be used for multiclass classification. The gradient is zero, which may cause problems in back propagation with a gradient descent method. This is why other activation functions are more common to use.

- **Linear:** Function shown in Figure 2.4b. In a linear activation function, f(x) = ax, the gradient is constant and independent of the input value. The error will not improve during training due to the same value for the derivative for every iteration, and therefore, there is no benefit in using the linear activation function,

- **Rectified linear unit (ReLU):** Function shown in Figure 2.4c and was proposed by Nair and Hinton in 2010 (Nair and Hinton, 2010) and has been the most widely used for deep learning applications (Chigozie Nwankpa and Marshall, 2018). ReLU learns faster than other non-linear activation functions such as the sigmoid function (LeCun et al., 2015) and allows training of deep supervised networks without unsupervised pretraining (Glorot et al., 2011). ReLU is widely used in convolutional neural networks and is used as the main activation functions such as ResNets (He et al., 2016a) and VGG nets (Simonyan and Zisserman, 2014).

- **Sigmoid:** Function shown in Figure 2.4d. The sigmoid function is a nonlinear and bounded function with outputs between 0 and 1. An advantage with the sigmoid function is that the derivative can be expressed in terms of itself (f'(x)=1-sigmoid(x)), making it computationally fast to derive the update equation when back propagation is used for training. Drawbacks with the sigmoid is that it suffers from slow convergence, gradient saturation and non-zero centered output (Chigozie Nwankpa and Marshall, 2018). A sigmoid function can also be used in the output layer for binary classification, since the output can be interpreted as probabilities between zero and one.

- **Softmax:** This activation function is often used in the output layer for multiclass classification networks to compute the probability distribution for the different classes, for example in the VGG networks (Simonyan and Zisserman, 2014). The softmax function is computed with the equation:

$$f(x_i) = \frac{e^{x_i}}{\sum_j x_j} \tag{2.3}$$

  The equation divides by the denominators of all activation functions in the output layer to get a probability distribution between zero and one.

The neurons in an ANN are normally connected in a layered structure, shown in Figure 2.3, where each neuron (the ellipses) is connected to each layer in the previous and the consecutive layer. The architecture consists of four main components:

- **Input layer:** There are the same number of neurons in the input layer as input variables. The inputs can be a feature vector or a measurement vector. The identity function is almost always used as the activation function for the input layer (Benardos and Vosniakos, 2007).
- **Hidden layers:** The number of hidden layers and the number of neurons in the layers can vary based on the complexity of the problem. Using more than one hidden layer will decrease the number of neurons needed. If it is necessary to use N hidden neurons to learn$N$ distinct samples (x_i, t_i), where $z_i \in R^n$ and $t_i \in R^m$, the network would become very large with increasing number of samples. In Huang (2003), Guang-Bin Huang proved that two two-hidden-layer feedforward networks (TLFNs) can be used with $L_1 = \sqrt{(m+2)N} + 2\sqrt{N/(m+2)}$ neurons and $L_2 = m\sqrt{N/(m+2)}$ neurons in the first and second hidden layers, respectively, can learn N samples with m output neurons, with any arbitrarily small error (Huang, 2003). A network with L_1 neurons in the first layer and L_2 neurons contains a total of $2\sqrt{(m+2)N}$ neurons, which is fewer neurons than a network with one hidden layer and N neurons. In addition, the TLFN grows with the square root of N, while the one-layered network grows linearly with N.
- **Output layer:** Responsible for producing the result, with one output neuron per output. In supervised learning, the output is evaluated against a desired or target response with a loss function.
- **Connections:** Connects the output from one node to the input of the nodes in the consecutive layer. Each connection has a weight that is updated and optimized during training.

2.1.2 Convolutional Neural networks

A convolutional neural network normally consists of two main parts. First is the convolution layers that learns to extract the important features from the image. Second is the fully connected layers that learn to classify the information received from the last convolution layer. Between the convolution layers and the fully connected layers is a flatten layer that reshapes the multidimensional output from the convolutional layers to an array of one dimension. This can be seen in Figure 2.5. After the flatten layer comes fully connected layers. This part of the CNN has the same structure as ANNs described in section 2.1.1.

Convolutional neural networks are a popular way of using machine learning to extract the desired information from images, especially in the later years when deep learning has gotten more attention. Deep learning is a branch within machine learning that uses several stacked layers of artificial neurons, which increases the complexity and abstraction. The development of graphic processing units (GPUs) has made it possible to develop fast deep neural nets by utilizing the number of cores in a GPU to do parallel computing. In Baykal et al. (2018), a desktop computer with Intel ® Core™ i5-6600 CPU with 4 cores is compared to a CUDA GPU architecture with 1152 cores. The CUDA architecture had a 9.65 faster execution time. In BUBER and DIRI (2018), a Tesla k80 GPU is compared with an Intel Xeon Gold 6126 CPU and the GPU had 4-5 faster execution time.

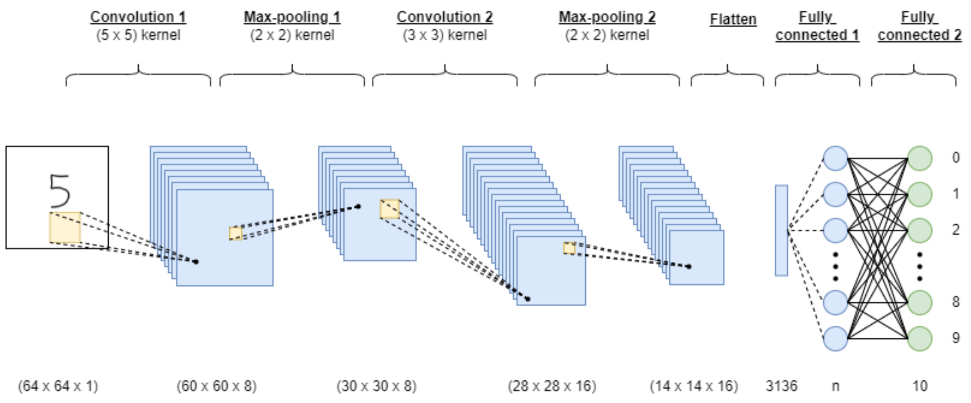
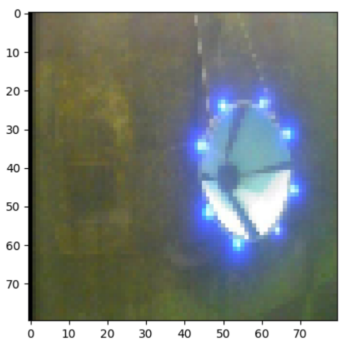
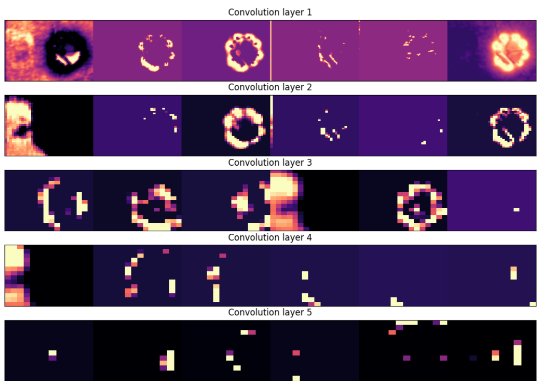


Figure 2.5: CNN for classification of hand drawn numbers

Figure 2.5 shows an example of a feed-forward CNN for classification of numbers and contains the most essential parts of a CNN. The input is a gray scale image with (64×64) pixels and the output is ten classes where each class represents the integers 0 to 9. The numbers below each layer describe the dimension and size of the layers.

Convolutional layer


(a) Input image form UDID dataset (Liu, 2018)



(b) Feature maps

Figure 2.6: Feature maps in a CNN for localization

The name convolutional neural network comes from the linear mathematical operation between matrices called convolution. The convolution layers are used to reduce the number of parameters before the fully connected layers, which is a beneficial aspect of a CNN (Albawi et al., 2017). The convolution layers contain kernels of size (3×3) or (5×5), but can also contain larger kernels. The convolution kernels act as filters that extract features from the image and create feature maps. Each convolution layer has several filters, and normally the number of filters increase, with the fewest filters in the first convolutional layer. In the training process, the kernel parameters are optimized to obtain important features from the image. It is necessary to use several convolution layers in a CNN because the deeper the input propagates through the convolution layers, the more abstract and complex shapes are extracted. Figure 2.6 shows five different convolution layers, where convolution layer one is the first and convolution layer five i the deepest. The CNN used in Figure 2.6 is trained to localize the underwater dock in the input image (Figure 2.6a by estimating a bounding box containing the dock. The first layer extracts the most basic shapes and filters out the background, and the fifth layer contains more abstract shapes. Here, the output from the fifth layer will be flattened and fed into fully connected layers.

Convolution works a weighted average operation over each point looked at. When looking at signal processing, convolution is a mathematical way of combining two signals to form a third signal. The two signals combined are the input signal and the impulse response, and they produce the output signal (Smith, 1997). This is a continuous operation and can be described with the equation,

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{2.4}$$

where $w(a)$, is the impulse response or weighing function, $x(a)$ is the input signal, $s(t)$ is the output signal, t is the time and a is the age of the signal (Goodfellow et al., 2016). In machine learning terminology, the weighing function w is referred to as the kernel or filter, and the output signal is referred to as the feature map. When working with convolution on computers with digital sensors, Equation 2.4 needs to be discretized, with w and x only defined on discrete times t:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.5}$$

In convolutional neural nets, the input is a multidimensional array of data, and the kernel is a multidimensional array of parameters that is updated during training. The infinite summation in Equation 2.5 can be implemented as a finite summation with the assumption that the input and kernel are zero everywhere except for the finite stored where the values for each element is stored (Goodfellow et al., 2016). Convolutions are often done over more than one axis at the time, so the equation for convolution with a two-dimensional image as input and a two-dimensional kernel becomes:

$$S[i,j] = (K * I)[i,j] = \sum_m \sum_n I[m,n]K[i-m, j-n] \tag{2.6}$$

here S in the new feature map I is the input image, K is the kernel, i is the row index of the input image, and j is the column index of the input image.

Since convolution is commutative, the equation can be equivalently as:

$$S[i,j] = (K * I)[i,j] = \sum_m \sum_n I[i-m, j-n]K[m,n] \tag{2.7}$$

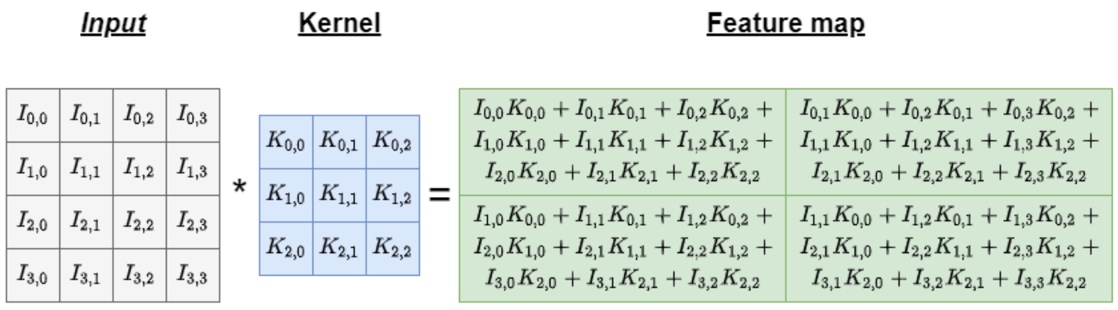

Input
Kernel
Feature map
$I_{0,0}$ $I_{0,1}$ $I_{0,2}$ $I_{0,3}$
$I_{1,0}$ $I_{1,1}$ $I_{1,2}$ $I_{1,3}$
$I_{2,0}$ $I_{2,1}$ $I_{2,2}$ $I_{2,3}$
$I_{3,0}$ $I_{3,1}$ $I_{3,2}$ $I_{3,3}$
$*$
$K_{0,0}$ $K_{0,1}$ $K_{0,2}$
$K_{1,0}$ $K_{1,1}$ $K_{1,2}$
$K_{2,0}$ $K_{2,1}$ $K_{2,2}$
$=$
$I_{0,0}K_{0,0} + I_{0,1}K_{0,1} + I_{0,2}K_{0,2} + I_{1,0}K_{1,0} + I_{1,1}K_{1,1} + I_{1,2}K_{1,2} + I_{2,0}K_{2,0} + I_{2,1}K_{2,1} + I_{2,2}K_{2,2}$
$I_{0,1}K_{0,0} + I_{0,2}K_{0,1} + I_{0,3}K_{0,2} + I_{1,1}K_{1,0} + I_{1,2}K_{1,1} + I_{1,3}K_{1,2} + I_{2,1}K_{2,0} + I_{2,2}K_{2,1} + I_{2,3}K_{2,2}$
$I_{1,0}K_{0,0} + I_{1,1}K_{0,1} + I_{1,2}K_{0,2} + I_{2,0}K_{1,0} + I_{2,1}K_{1,1} + I_{2,2}K_{1,2} + I_{3,0}K_{2,0} + I_{3,1}K_{2,1} + I_{3,2}K_{2,2}$
$I_{1,1}K_{0,0} + I_{1,2}K_{0,1} + I_{1,3}K_{0,2} + I_{2,1}K_{1,0} + I_{2,2}K_{1,1} + I_{2,3}K_{1,2} + I_{3,1}K_{2,0} + I_{3,2}K_{2,1} + I_{3,3}K_{2,2}$

Figure 2.7: Example of discrete convolution

Equation 2.7 is normally more convenient to implement in a CNN because of less variation of valid integers of m and n. Figure 2.7 shows an example of Equation 2.7 where the input image is (4 × 4), the kernel is (3 × 3) and the feature map becomes (2 × 2). This convolution is done with stride 1, which means the kernel convolves around the input image by shifting one unit at a time. A larger stride will give a lower output resolution. The output has two fewer columns than the input image. This can be avoided by using padding on the input image. A zero padding is popular to use in convolutional neural networks to maintain the feature map size and to reduce the information bias against the boundary (Alsallakh et al., 2020). Maintaining the feature map size can be important to handle arbitrary input sizes, maintaining aspect ratio for non-square input images. This is also important when concatenating feature maps from different layers, which is often used in image segmentation, for example in the network U-Net (Ronneberger et al., 2015). Information bias can be seen in Figure 2.7. The corner pixels are convoluted one time (they appear each in one of the feature map pixels), the edge pixels are convoluted two times, while the center pixels are convoluted four times. A same zero padding means, for a (3×3) kernel, adding an outer frame with width one around the input image. A padding width, w for an arbitrary kernel with size, (kxk) can be found by:

$$w = \frac{k}{2} - 0.5 \tag{2.8}$$

when k is an odd number. The kernel size is rarely an even number when dealing with standard CNNs. Generative adversarial networks (GANs) often use kernels with size (4×4) in combination with a stride of two to avoid checkerboard artifacts (Miyato et al., 2018). A convolution layer will reduce the number of trainable parameters drastically compared to a fully connected layer, without loss of performance in the process. This can be done by utilizing the sparse connectivity of an image. With a (3×3) kernel, only 9 units in the next layer are affected by a unit from the input layer. The receptive field of one unit will increase with the dept of the convolution layers. This means that units in the deeper layers can be indirectly connected to all the pixels in the input image With (3×3) convolution kernels. Nine trainable parameters are needed to do convolution for one unit in the next layer. In convolution, parameter sharing is used, so the same 9 parameters are used for all the units in one feature map. Normally, one additional parameter is used, a bias, that is independent of the input. As an example, one layer has the dimension (32 × 32 × 3) and the next layer has the dimension (30 × 30 × 6). By using a fully connected layer, the number of parameters, Pfc would be:

$$P_{fc} = W_i \cdot H_i \cdot D_i \cdot W_j \cdot H_j \cdot D_j + B_{fc} = 32^2 \cdot 3 \cdot 30^2 \cdot 6 + \cdot 30^2 \cdot 6 = 16594200 \tag{2.9}$$

where W_i is the width of the first layer, H_i is the height of the first layer, D_i is the depth of the first layer, w_j is the width of the second layer, H_j is the height of the second layer, D_j is the height of the second layer and B_{fc} is bias, equal to neurons in the second layer. When using a convolutional layer with a kernel size K = 3, the number of parameters, P_c, would be:

$$P_c = K^2 \cdot D_i \cdot D_j + B_c = 3^2 \cdot 3 \cdot 6 + 6 = 168 \tag{2.10}$$

where B_c is bias in the convolution layer, and there is one bias per kernel. There are as many kernels as the depth of the output. As can be seen from Equation 2.9 and Equation 2.10, $Pc \ll Pfc$ and will decrease the run-time of forward propagation and decrease the storage capacity needed. The sparse connectivity will reduce the runtime and the parameter sharing will reduce the storage capacity needed (Goodfellow et al., 2016).

Pooling Layer

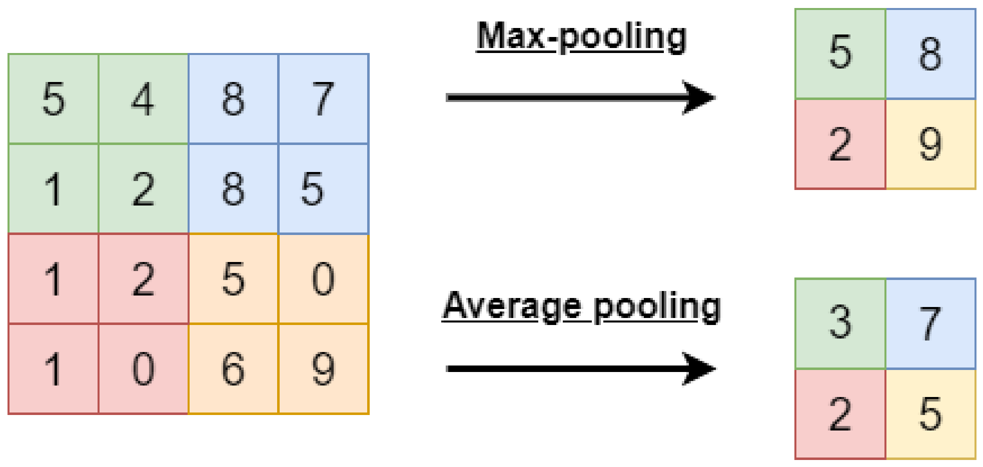


Figure 2.8: Max-pooling and average pooling on a feature map

Almost all convolutional networks apply pooling layers. This is normally done after each convolution layer or after a block of convolution layers. The convolution produces a set of linear activations. Between the pooling layer and the convolution layer is a nonlinear activation layer that each linear activation is passed through to capture and learn the non-linearities (Scherer et al., 2010). ReLU is the most common to use and is described in section 2.1.1. The purpose of pooling is to reduce the resolution of the feature maps to achieve spatial invariance and invariance of small translation of the input. Reducing the resolution will also result in fewer parameters and will reduce the statistical and computational burden on the next layer (Goodfellow et al., 2016). Pooling combines an (nxn) patch of units into a new unit. With $n = 2$ the pooling will half the resolution of the input. Each pooled feature map corresponds to one feature map from the previous layer. Some popular pooling methods are average pooling, max-pooling, mixed pooling, L_p pooling, stochastic pooling, spatial pyramid pooling and region of interest pooling (Gholamalinejad and Khosravi, 2020). Max-pooling and average pooling is shown in Figure 2.8, where a (4×4) input is reduced to a (2×2) output. Normally, the max-pooling is used because it will extract the most prominent features from the previous feature-map.

Flatten Layer

A flatten layer reshapes the tensor from a multidimensional tensor, with shape ($height$, $width$, $number\ of\ filters$), to a one-dimensional tensor with $length = height \cdot width \cdot number\ of\ filters$. The flatten layer is the connection between the convolutional part of the CNN and the fully connected part of the CNN. The fully connected layers are applied to learn non-linear combinations from the features extracted by the convolutional layers. The last layer of the fully connected part represents the output of the network and normally has the architecture shown in Figure 2.3.

Other layer structures

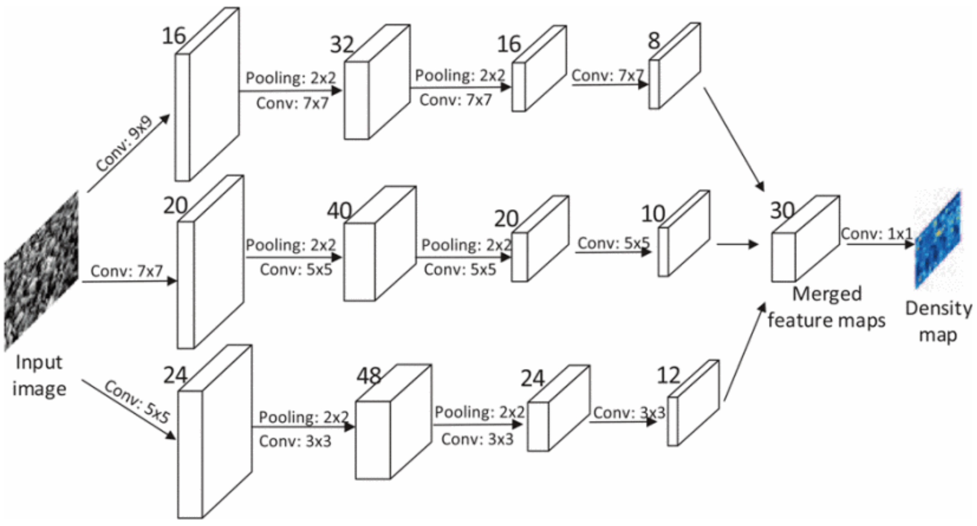


Figure 2.9: Multi-column structure. Courtesy: Zhang et al. (2016)

Figure 2.9 shows a network structure that has three parallel feed-forward layers. Each column has a different filter size, and this can be used to capture perspective changes and image resolution changes. The specific network shown in Figure 2.9 is used for single-image crowd counting, where the different columns will learn to detect heads of different sizes (Zhang et al., 2016). According to Zhang et al. (2016), this multi-column neural network model will outperform existing models because of its ability to handle arbitrarily perspectives and resolutions.

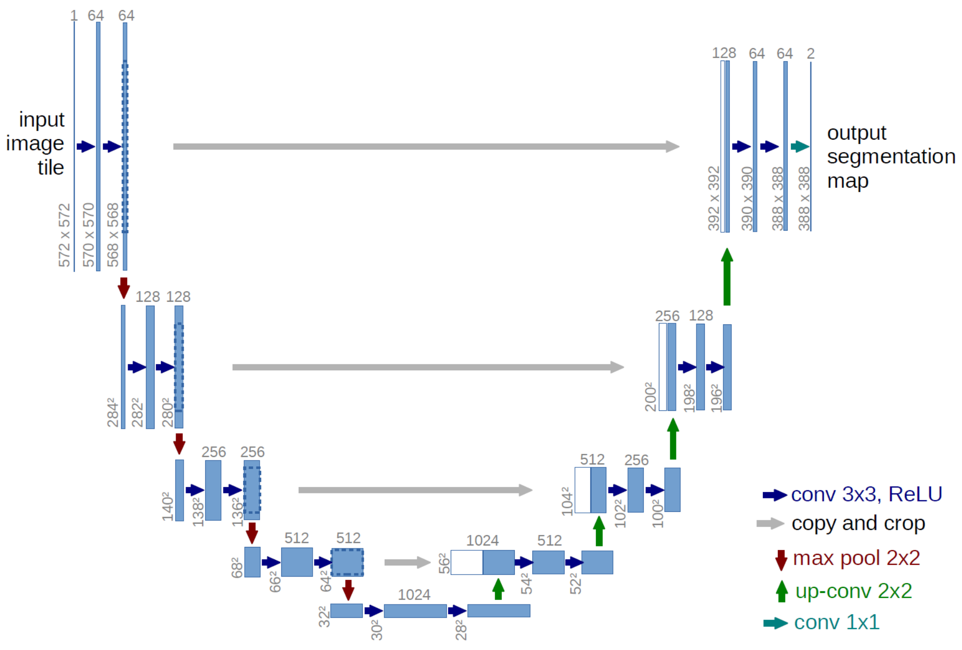


Figure 2.10: U-Net. Courtesy: Ronneberger et al. (2015)

Figure 2.10 shows a convolutional neural network where a skip connection feeds the output of one layer to layers later in the networks. This means that some layers will have several inputs from different layers.

The most common way of combining the layers is by addition, as used in ResNnet (He et al., 2016b), and concatenation, used in U-Net(Ronneberger et al., 2015). Skip connections are commonly used in CNNs that create images, such as GANs and semantic segmentation networks. The skip connections is used to pass low-level information across a network to rebuild an image.

2.1.3 Recurrent Neural Networks

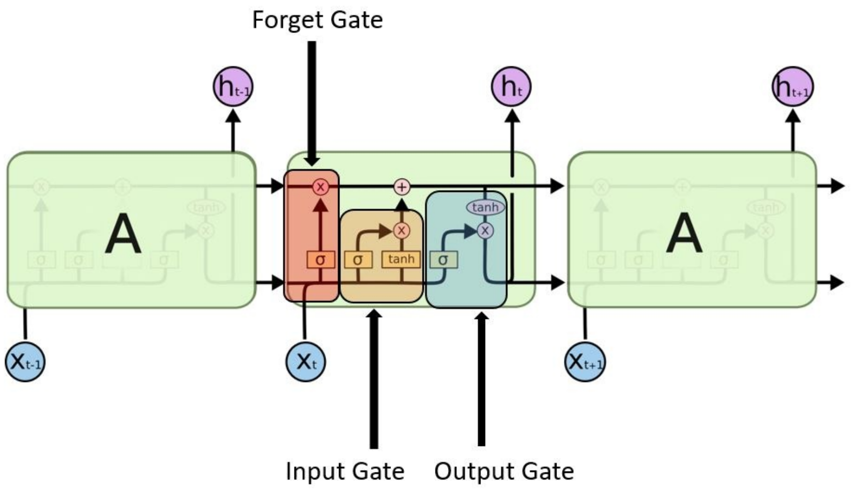


Figure 2.11: LSTM structure. Courtesy: Kim et al. (2021)

Recurrent neural networks (RNNs) are ANNs with feedback connections. RNNs are often used in speech recognition, and other cases where the order of the sequence will matter on the results. For example, if the use case is speech recognition and the current predicted word is “of”, the most probable following word is “the” (Umeda and Kahn, 1981). This information can be exploited for better predictions. Figure 2.11 shows a popular block used for RNNs. The block is called a long short-term memory (LSTM) block. The LSTM block contains an input gate, output gate and forget gate. Values will be remembered over an arbitrary time interval, where the three gates will regulate the flow of information.

2.2 Semantic Segmentation

A convolutional neural network can have many use cases. The most popular use cases are classification, where each image has a label, localization, where each image has normally four labels for x-position, y-position, width and height to form a bounding box around the localized object, and segmentation, where each image has as many labels as the input image has pixels. The structure of Figure 2.5 is normally used for labeling, localization and object detection, object detection is localization of classes, where more than one class can be localized. A structure like Figure 2.5 and be used for segmentation tasks too, but fully convolutional networks (FCNs) have been proven to exceed the state-of-the-art (SOTA) CNNs for semantic segmentation from supervised training (Long et al., 2015).

2.2.1 Fully Convolutional Networks

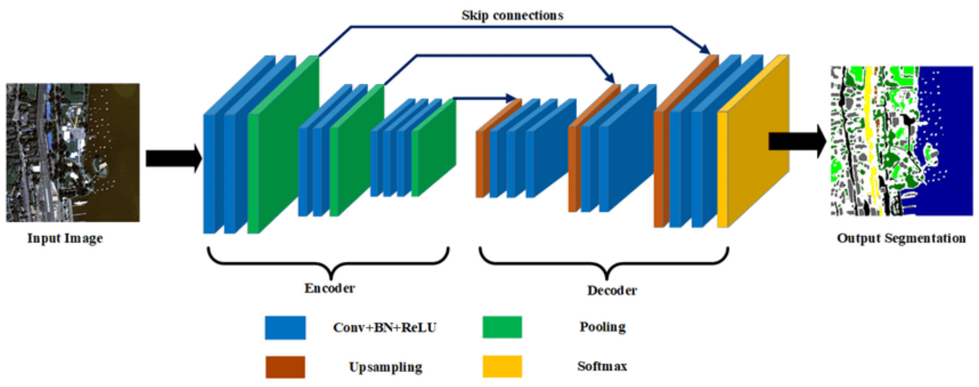


Figure 2.12: Architecture of a fully convolutional network. Courtesy: Peng et al. (2019)

A CNN with a convolutional part and a dense part (Figure 2.5) will normally produce non-spatial outputs with a fixed size input. One can look at the fully connected layers as 1×1 convolutions (Long et al., 2015). Changing from fully connected layers to 1×1 convolutions will change the network to a fully convolutional network that can take input of any size and output spatial classification maps. In addition, changing from dense layers to convolution layers will make the network five times faster than before the transformation (Long et al., 2015).

A network for semantic segmentation needs to learn both the semantics and the location of the classes. A FCN takes advantage of the way the multi-layered convolutional network extracts information from an image, described in section 2.1.2. The deep layers extract semantic information, while the first layers extracts appearance and localization information, as shown in Figure 2.6b.A way to utilize this is by the use of skip-connections through the network. This can be seen in Figure 2.10, where the output of the first layer is connected as part of the input in the last layer, to give coarse, spatial information about the localized classes to the last layer, which also receives coarser semantic information from the deeper layers. A FCN can take an input of any size and the output has corresponding spatial dimensions as the input. A FCN is faster than a conventional CNN, but to increase the computational efficiency, the feature maps are normally down-sampled at each layer by using a max-pooling step, described in section 2.1.2. To produce output maps with equal size as the input image, the feature-maps is up-scaled with up sampling or deconvolution. Up sampling is the opposite of pooling, while deconvolution is the opposite of convolution.

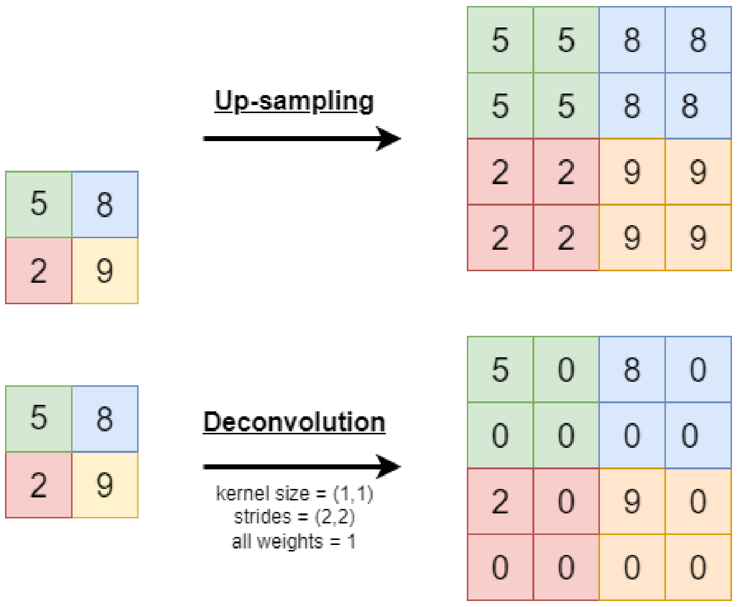


Figure 2.13: Deconvolution and up-sampling

An illustration of how up-sampling and deconvolution is calculated can be seen in Figure 2.13. If pooling is done on the output of the up-sampling, one get the input to the up sampling. The same applies for deconvolution and convolution. The deconvolution contain trainable parameters, so it will improve during training. Since the up-sampling does not have any trainable parameters, a convolution layer is normally placed after the up-sampling to interpret and fill in the up-sampled feature map. The down-sampling part of the FCN can be seen as a decoder of the image, and the up-sampling part is the encoder. An FCN consisting of a decoder and encoder is a common architecture for a semantic segmentation model. The decoder is similar to the convolutional part in a conventional CNN, so popular CNNs are normally used for the decoding, where the CNNs for decoding can be seen as the backbone of the FCN. The decoder is mapping from pixels to features and is aimed at finding context, or semantic information. The decoder uses information from different stages of the encoder to transform the feature maps to pixel maps and recovering spatial information. Figure 2.12 shows how a FCN architecture can look like, where the encoder and decoder can be seen as mirrored versions of each other. The FCN shown uses Softmax activation in the output layer to estimate the probability of a pixel belonging to a certain class. If each pixel only can be one class, argmax can be used on each pixel and choose the class with the highest probability. In a case where classes can overlap, a positive recognition can be determined by a 0.5 threshold of the Softmax output. The label is normally one-hot encoded, which means that the network will have as many output masks as there are classes to predict and the same width and height as the input image.

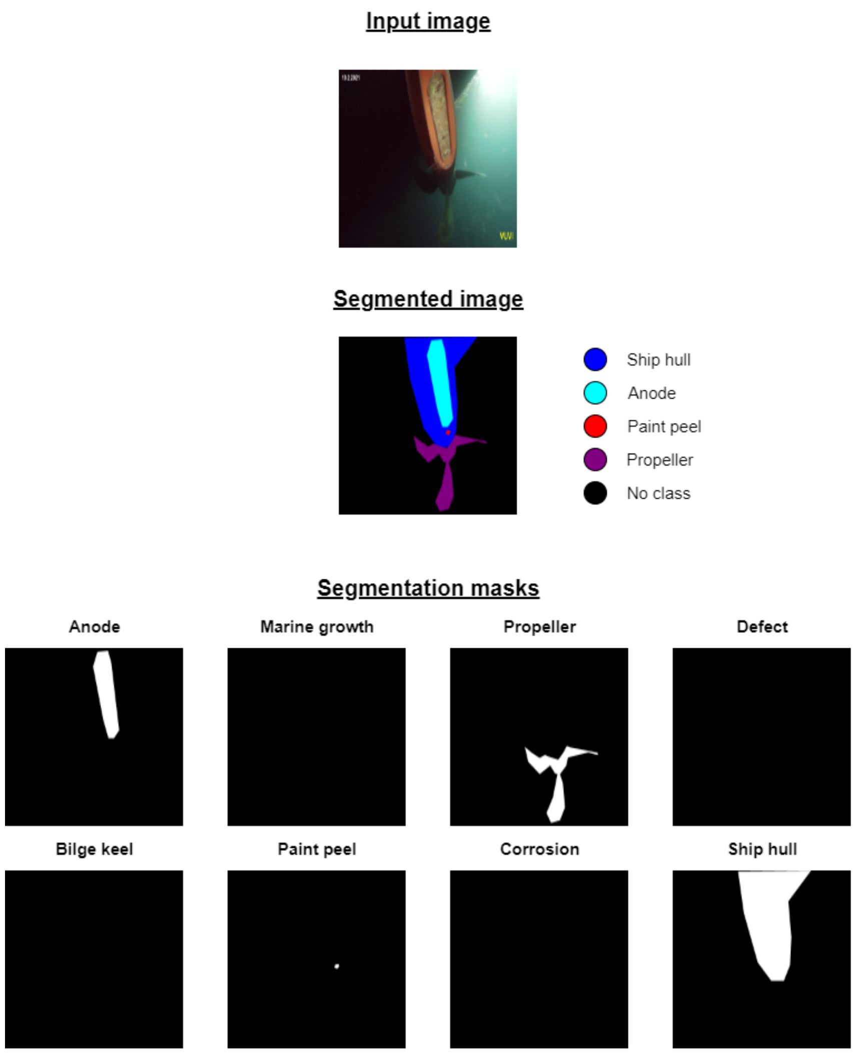


Figure 2.14: Input and labels for semantic segmentation

Figure 2.14 shows how an input image is related to the labels and the segmented ground truth, with eight different classes. The image is an RGB-images, so the input will have a depth of three. The output consists of eight binary masks, where the pixels will have value one where the classes are present and zero where the classes are not present. There will be one output of the FCN for each of the masks, and the output have the same width and height as the input image.

2.2.2 Training

The main concept behind supervised learning is to find a generalized mapping from the input to the target output:

$$Y = f(X) \tag{2.11}$$

Where X is the input and Y is a set of targets. The aim during training is to find a function, $hatf$, such that $\hat{Y} = f(\hat{X})$ is close to the target Y. The input and output can be vectors or matrices for higher order. With red, green, blue (RGB) images as input, the input has a shape of (p_xxp_y 3) with p_x pixels in width and p_y pixels in height and tree color channels. The output vector will have the shape (1×c) for image classification (Tripathi and Kumar, 2019), (4×c) for object localization (Liu et al., 2018) or (p_xxp_y×c) for segmentation, where c is the number of classes. The training of the neural network consists of two parts. Forward propagation and back-propagation. In the forward propagation, a prediction of the response based on the input is made. The loss between the predicted and actual response is then back-propagated from the output layer to the hidden layers. In the back-propagation, the loss is minimized by an optimization algorithm based on the loss rate from the previous epoch. Since the loss rate is needed in the back-propagation, the loss function needs to be differentiable. Common loss functions are mean squared error and binary cross-entropy loss and categorical cross-entropy loss.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{2.12}$$

$$BCE = -\frac{\sum_{i=1}^{n} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)}{n} \tag{2.13}$$

$$CCE = -\sum_{i=1}^{n} y_i log(\hat{y}_i) \tag{2.14}$$

$$FL = -\sum_{i=1}^{n}(1 - \hat{y}_i)^{\gamma} y_i log(\hat{y}_i) \tag{2.15}$$

Equation 2.12, Equation 2.13 and Equation 2.14 shows the formula for mean squared error, binary cross entropy and categorical cross entropy. Binary cross-entropy can be used in binary classification where the target label is 0 or 1. Categorical cross entropy is used in multi-class classification, where the target labels need to be one-hot encoded. Focal loss (2.15), introduced by Lin et al. (2017) is a loss function that reduces the relative loss for well classified examples for increasing γ. The focal loss adds the factor $(1 - \hat{y}_i)^{\gamma}$ to the standard cross entropy function, and γ = 0 gives the cross entropy. Focal loss aims to improve the learning when dealing with class imbalances.

During the back propagation, an optimizer is used to minimize the loss function. One common optimizer is the gradient descent method, where each weight in the neural network is updated iteratively based on the gradient of the cost function. To get the gradient of the cost function with regard to each weight, back propagation utilizes the chain rule to calculate each gradient one layer at the time, starting from the output layer.

The gradient descent can be described with the equations

$$\Delta S = \left(\frac{\partial S}{\partial w_{11}}, \frac{\partial S}{\partial w_{12}}, ...\right)^T \tag{2.16}$$

$$w_{i,j}^{n+1} = w_{i,j}^{n} - \alpha(\Delta S)_{i,j} \tag{2.17}$$

where S is the cost function, w is the weights, n is the iteration number, i is the layer number, j is the node number and α is the step size, which can be seen as the learning rate of the training process. Too small α leads to slow convergence, and too large α can lead to oscillations and instability in the learning process. In (2.17), the weight update using the gradient descend method is described. Gradient descent can cause problems in the learning since it can get stuck at a local minimum. Adding momentum will help to avoid this, by adding a parameter to let the optimizer to not stop directly at a minimum, but continue for a few iterations by using an exponentially weighted average. gradient descent with momentum can be described with the equation,

$$w^{n+1} = w^{n} - \alpha m_n \tag{2.18}$$

where m_n is the aggregate of gradients at iteration n:

$$m_n = \beta m_{n-1} + (1-\beta)(\Delta S) \tag{2.19}$$

where β is the moving average parameter, often set to 0.9.

Adam (Kingma and Ba, 2014) is an alternative optimizer for first-order gradient-based optimization. Adam require little memory and is efficient, and therefore well-suited for large-scale training such as CNN training. Adam can be seen as a combination of gradient descent with momentum (2.18) and the root mean square propagation optimizer, which is based on the exponential moving average of the gradient:

$$w_{n+1} = w_n - \frac{\alpha_t}{\sqrt{v_n + \epsilon}} \Delta S \tag{2.20}$$

where ϵ is a small number to avoid dividing by zero, $(\cdot_n)$ denotes the current iteration and v is the sum of the square of past gradients:

$$v_n = \beta v_{n-1} + (1-\beta)(\Delta S)^2 \tag{2.21}$$

Adam combines these equations:

$$\begin{aligned} m_n &= \beta_1 m_{n-1} + (1-\beta_1)(\Delta S) \\ v_t &= \beta_2 v_{n-1} + (1-\beta)(\Delta S)^2 \end{aligned} \tag{2.22}$$

where β_1 and β_2 are the decay rates for the methods combined in Adam, normally set to 0.9 and 0.999, respectively. Since the decay rates are close to one m_n and v_n tends to be biased towards zero. m_n and v_n are therefore bias-corrected:

$$\begin{aligned} \hat{m}_n &= \frac{m_n}{1-\beta_1} \\ \hat{v}_n &= \frac{v_n}{1-\beta_2} \end{aligned} \tag{2.23}$$

where $\hat{m}_n$ is the bias-corrected version of m_n and $\hat{v}_n$ is the bias corrected version of v_n. Adam combines the bias corrected equations (2.23) and (2.22) with the equation:

$$w_{n+1} = w_n - \hat{m}_n \left(\frac{\alpha}{\sqrt{\hat{v}_n + \epsilon}} \right) \tag{2.24}$$

The supervised training is done with a dataset containing images and corresponding labels. The network is fed batches of images, which are used to make predictions to match the labels. The network is fed batches to decrease the training time, but also to keep the network more generalized. After all the images from the dataset are fed through the network, one epoch is done and a new epoch is started by feeding the dataset through the network again. The complete dataset is normally divided into three different datasets: one training dataset, one validation dataset and one testing dataset. The training set is used for training and updating the network. The validation set is used to validate the performance from the network after each epoch. The performance can be defined as the loss or another metric, such as mean squared error, accuracy, dice loss or Jaccard loss. The validation set can be used to determine if the weights from the network shall be saved after an epoch is finished based on if the performance on the validation set have improved. The test set is used after the training to cross validate the best weights from the training on an independent data set. The training set is normally 60 to 80% of the complete set, and the validation and test set can be 5 to 15 % of the complete set each.

Overfitting

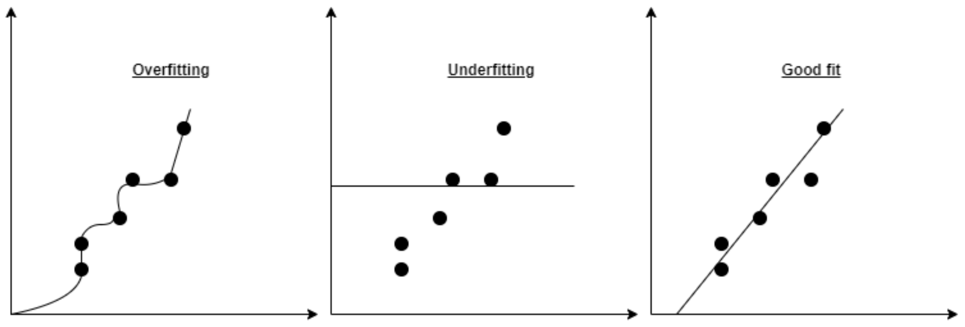


Figure 2.15: Example of overfitting

As mentioned in section 2.2.2, it is desired to acquire a generalized mapping from the input to the labels. As can be seen in Figure 2.15, an overfitted model will achieve a very good result on the set, but not be able to handle other inputs well. That is why the validation set is used to determine when to save the model. Overfitting happens when the performance on the test set continues to improve at each iteration, while the performance on the validation set will stagnate or decrease at each iteration.

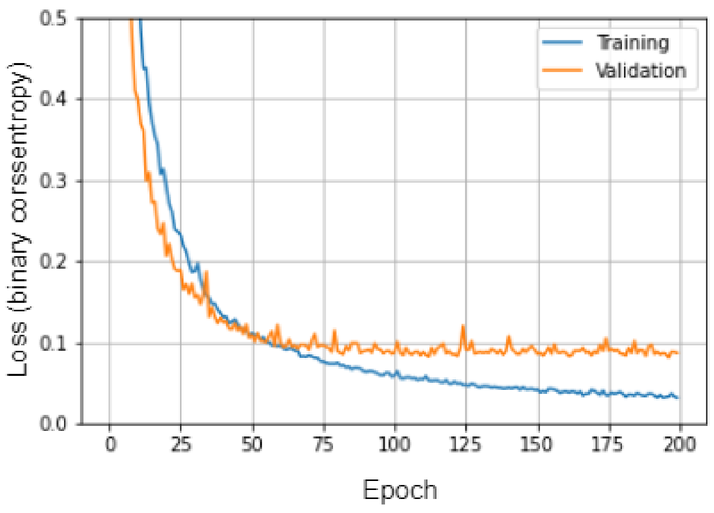


Figure 2.16: Overfitting during training

Figure 2.16 shows how the model can start overfitting during training. The performance on the test set will continue to improve, while the performance on the validation set will stagnate after approximately 100 epochs. To decrease the level of overfitting, several measures can be taken. The images and corresponding masks can be shuffled between each epoch, so the network will not learn any contexts between subsequent images.

Another way of reducing the overfitting is by augmenting the input images. This can be done before the learning process to expend the dataset, or it can be done continuous each time a new batch is loaded with random augmentation variables with a given probability of applying the augmentation. This can create an almost infinite number of variations of the input image and keep the network to learn too specific details of the input image. The augmentations can be chosen to create realistic variations of the input image to increase the prediction capabilities in different environments than what is represented in the training set. It is important to apply the same augmentations on the corresponding masks if the augmentation changes the spatial information in the image.

The fully convolutional neural network can also be changed to be more generalizing for overfitting avoidance. One common technique is to apply dropout between each layer. Dropout will cause a connection to be set to zero with a certain probability. Dropout is most common used in fully connected neural nets, but has also proven to be effective in convolutional neural nets, especially the dropout type, channel dropout (Spilsbury and Camps, 2019), where the connections for entire feature maps is set to zero instead of randomly selected single connections. In CNNs, however, batchnormalization (Ioffe and Szegedy, 2015) is the most common way to reduce overfitting. Batchnormalization will bring the mean to zero and the variance to one, which will regularize the model and speed up the training. Batchnormalization will also enable higher learning rates.

Transfer Learning

Transfer learning is used to speed up and improve the learning process. Before the training begins, weights from a trained network are loaded to the network to be trained. The pretrained network is normally trained on a larger dataset than the current training shall be done on. For semantic segmentation, the encoder weights are especially relevant to use in transfer learning, since encoders are normally standard CNNs such as VGG16, VGG19, MobileNet, ResNet and Inception. The encoder is working as a feature extractor, so whether the pretrained network is trained on other classes or other recognition types, the way a net is extracting features will be similar to how the current network need to extract features. Using transfer learning is also helpful when using dropout, since dropout can cause divergence in the beginning of the learning process when no features are learned. The decoder weights normally need to be more specialized for each case, so transfer learning from other cases is not so useful here.

2.2.3 Evaluation Metrics

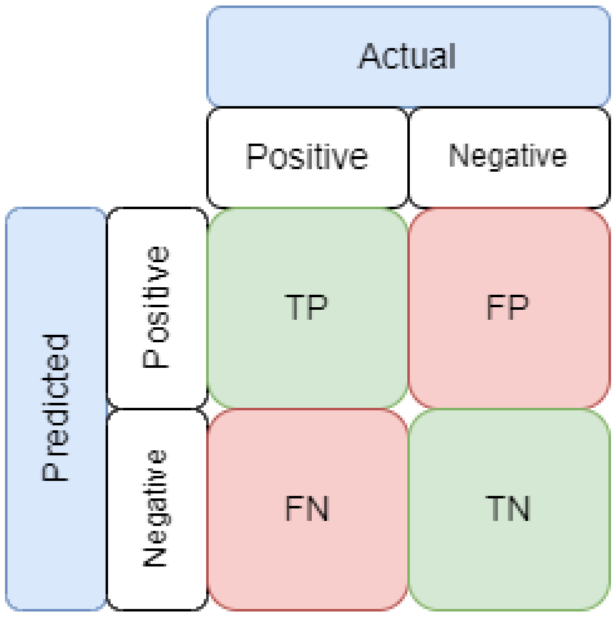


Figure 2.17: Basic evaluation metrics

When evaluating the performance of the trained semantic segmentation models, the predictions can be classified into four categories: True positives(TP), true negatives(TN), false positives(FP) and false negatives(FN), as shown in Figure 2.17. These four metrics give all the information about the performance, but they need to be used together to get a general and comparable metric.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.25}$$

Accuracy (2.25) is one metric that gives the percentage of correctly pixel predictions. Accuracy can work well when half of the label values are positive, but when, for example, only 10% are positive, the network would get an accuracy score of 90% if all predictions are negative, which is not a good performance indicator in general. There exists other metrics that takes into account the ratio of positive labels:

$$Precision = \frac{TP}{TP + FP} \tag{2.26}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.27}$$

Precision and recall (2.26), (2.27) are useful then evaluating a CNN trained on a dataset with a relatively low or high ratio of positive labels of each class. A system with high precision and low recall means too few positive predictions, but the positive predictions are mostly correct. A high recall and low recall means many correct predictions, but also many false positives, which means too many positive predictions. A high precision and a high recall is therefore desired.

$$F_1 = \frac{1}{\frac{1}{2}\left(\frac{1}{Recall} + \frac{1}{Precision}\right)} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.28}$$

To evaluate the precision and recall, another metric can be used, which is a measurement of the trade-off between the precision and recall. F_1 score, or dice score, uses the harmonic average of precision and recall. Harmonic average gives the average when two rates are considered, and is also a measure to handle outliers and punish extreme values. For example, with a precision of 0.9 and recall of 0.1, the arithmetic mean is 0.5 while the harmonic mean is 0.18.

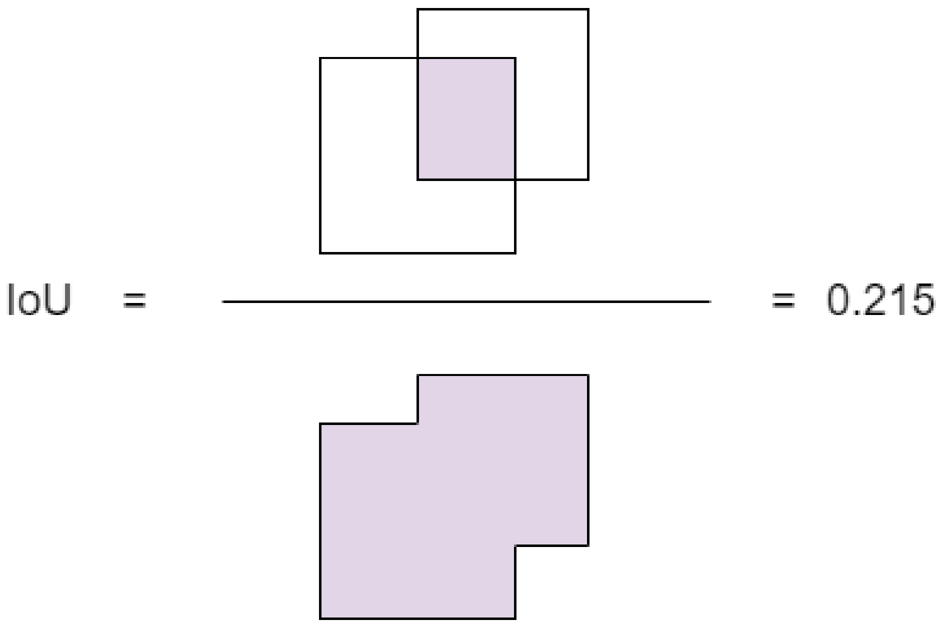


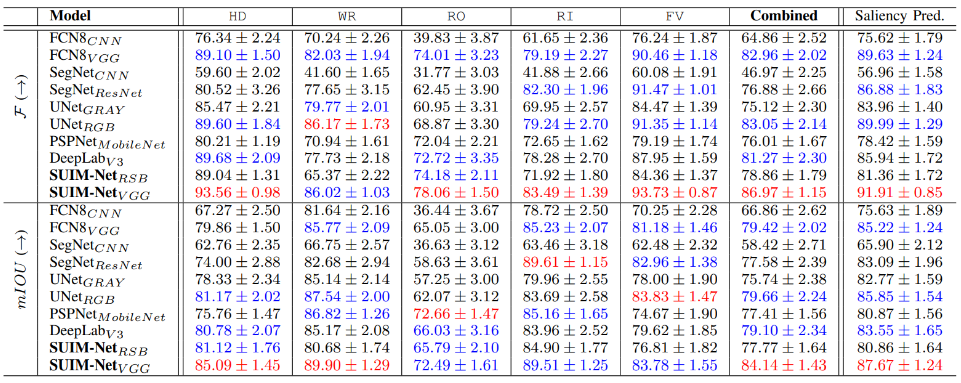
Figure 2.18: Intersection over union

$$IoU = \frac{Overlap}{Union} = \frac{TP}{TP + FP + FN} \tag{2.29}$$

The intersection over union (IoU) score, or Jaccard index (2.29) is a ratio where the overlapping area of positives from the predictions and labels are divided by the total area of positives in the predictions and labels. A visual representation of the Iou score is shown in Figure 2.18 the Iou score gives a good visual representation of the performance. IoU and F_1 score are based on different concepts, but what they represent are similar. They are positivety correlated with F_1 always larger or equal than IoU. The IoU score is always larger or equal than half of the F_1 score: $\frac{F_1}{2} \leq IoU \leq F_1$.

2.2.4 Existing Models

There exists many open-source models for semantic segmentation, The advantage of FCNs for semantic segmentation is that they are general and often simple to alter to specific tasks. In Mo et al. (2022), several of the SOTA models are compared. In Kamann and Rother (2020), different models and backbones are tested on different degraded images from datasets such as Cityscapes (Cordts et al., 2016), a dataset for semantic segmentation in an urban environment.



	Model	HD	WR	RO	RI	FV	**Combined**	Saliency Pred.
$\mathcal{F}$ (↑)	FCN8$_{CNN}$	76.34 ± 2.24	70.24 ± 2.26	39.83 ± 3.87	61.65 ± 2.36	76.24 ± 1.87	64.86 ± 2.52	75.62 ± 1.79
	FCN8$_{VGG}$	89.10 ± 1.50	82.03 ± 1.94	74.01 ± 3.23	79.19 ± 2.27	90.46 ± 1.18	82.96 ± 2.02	89.63 ± 1.24
	SegNet$_{CNN}$	59.60 ± 2.02	41.60 ± 1.65	31.77 ± 3.03	41.88 ± 2.66	60.08 ± 1.91	46.97 ± 2.25	56.96 ± 1.58
	SegNet$_{ResNet}$	80.52 ± 3.26	77.65 ± 3.15	62.45 ± 3.90	82.30 ± 1.96	91.47 ± 1.01	76.88 ± 2.66	86.88 ± 1.83
	UNet$_{GRAY}$	85.47 ± 2.21	79.77 ± 2.01	60.95 ± 3.31	69.95 ± 2.57	84.47 ± 1.39	75.12 ± 2.30	83.96 ± 1.40
	UNet$_{RGB}$	89.60 ± 1.84	86.17 ± 1.73	68.87 ± 3.30	79.24 ± 2.70	91.35 ± 1.14	83.05 ± 2.14	89.99 ± 1.29
	PSPNet$_{MobileNet}$	80.21 ± 1.19	70.94 ± 1.61	72.04 ± 2.21	72.65 ± 1.62	79.19 ± 1.74	76.01 ± 1.67	78.42 ± 1.59
	DeepLab$_{V3}$	89.68 ± 2.09	77.73 ± 2.18	72.72 ± 3.35	78.28 ± 2.70	87.95 ± 1.59	81.27 ± 2.30	85.94 ± 1.72
	SUIM-Net$_{RSB}$	89.04 ± 1.31	65.37 ± 2.22	74.18 ± 2.11	71.92 ± 1.80	84.36 ± 1.37	78.86 ± 1.79	81.36 ± 1.72
	SUIM-Net$_{VGG}$	93.56 ± 0.98	86.02 ± 1.03	78.06 ± 1.50	83.49 ± 1.39	93.73 ± 0.87	86.97 ± 1.15	91.91 ± 0.85
$mIOU$ (↑)	FCN8$_{CNN}$	67.27 ± 2.50	81.64 ± 2.16	36.44 ± 3.67	78.72 ± 2.50	70.25 ± 2.28	66.86 ± 2.62	75.63 ± 1.89
	FCN8$_{VGG}$	79.86 ± 1.50	85.77 ± 2.09	65.05 ± 3.00	85.23 ± 2.07	81.18 ± 1.46	79.42 ± 2.02	85.22 ± 1.24
	SegNet$_{CNN}$	62.76 ± 2.35	66.75 ± 2.57	36.63 ± 3.12	63.46 ± 3.18	62.48 ± 2.32	58.42 ± 2.71	65.90 ± 2.12
	SegNet$_{ResNet}$	74.00 ± 2.88	82.68 ± 2.94	58.63 ± 3.61	89.61 ± 1.15	82.96 ± 1.38	77.58 ± 2.39	83.09 ± 1.96
	UNet$_{GRAY}$	78.33 ± 2.34	85.14 ± 2.14	57.25 ± 3.00	79.96 ± 2.55	78.00 ± 1.90	75.74 ± 2.38	82.77 ± 1.59
	UNet$_{RGB}$	81.17 ± 2.02	87.54 ± 2.00	62.07 ± 3.12	83.69 ± 2.58	83.83 ± 1.47	79.66 ± 2.24	85.85 ± 1.54
	PSPNet$_{MobileNet}$	75.76 ± 1.47	86.82 ± 1.26	72.66 ± 1.47	85.16 ± 1.65	74.67 ± 1.90	77.41 ± 1.56	80.87 ± 1.56
	DeepLab$_{V3}$	80.78 ± 2.07	85.17 ± 2.08	66.03 ± 3.16	83.96 ± 2.52	79.62 ± 1.85	79.10 ± 2.34	83.55 ± 1.65
	SUIM-Net$_{RSB}$	81.12 ± 1.76	80.68 ± 1.74	65.79 ± 2.10	84.90 ± 1.77	76.81 ± 1.82	77.77 ± 1.64	80.86 ± 1.64
	SUIM-Net$_{VGG}$	85.09 ± 1.45	89.90 ± 1.29	72.49 ± 1.61	89.51 ± 1.25	83.78 ± 1.55	84.14 ± 1.43	87.67 ± 1.24

Figure 2.19: Quantitative performance comparison of different semantic segmentation models. Courtesy: Islam et al. (2020a)

In Figure 2.19 several SOTA models for semantic segmentation are compared with the quantitative metrics IoU and F_1 score. The models compared are FCN8, SegNet, UNet, PSPNet, DeepLab and SUIM-Net. The models are trained on the semantic Segmentation of Underwater IMagery (SUIM) dataset, proposed in Islam et al. (2020a). The SUIM dataset is the first large-scale annotated dataset for general-purpose semantic segmentation of underwater scenes (Islam et al., 2020a). The dataset contains 1635 underwater images with corresponding segmentation mask with 8 different pixel annotations. Since this performance comparison is done on an underwater dataset, it is probable that these results will be more relevant to other underwater segmentation cases, than comparison studies done on for example Cityscapes. In Kamann and Rother (2020), the models performs unequal on different types of corrupted images. For example, have Xception-41 the highest average mean IoU on four of the five noise corruption methods tested, but only have the highest mean IoU on two of the remaining 15 tests done. Underwater images often have the same type of image corruptions, as commented in section 2.3 and is therefore more comparable.

2.2.5 Encoders

Existing CNNs for classifications are often used as encoders, or backbones, to FCNs. The fully connected part of the existing CNNs are removed to fit in a FCN.

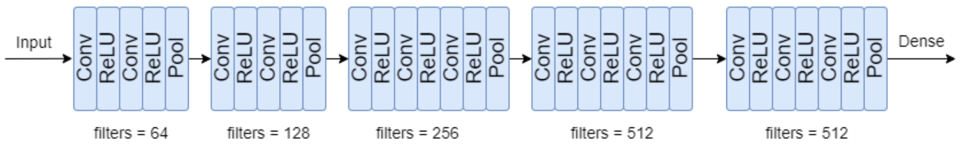


Figure 2.20: VGG16

The VGG16 (Simonyan and Zisserman, 2014) CNN was proposed in 2013. VGG16 is shown in Figure 2.20 and consists of (3×3) convolutions, ReLU activations and max pooling layers. The filter number doubles in each block, while the resolution (height $\times$ width) halves. The full VGG16 with the fully connected part consists of a 138 million weight parameters, and with only the convolutional part, the model consists of 15 million trainable weights. MobileNets (Howard et al., 2017) are models created for efficiency for mobile and embedded applications. They based on a streamlined architecture with depth-wise separable convolutions.Depth-wise convolutions applies one convolution filter per input channel. Then a pointwise convolution with shape (1×1×input channels), is used to create a linear combination of the output from the depth-wise convolution. Standard convolution performs a spatial-wise and channel-wise convolution in one step. Standard convolution have a computational cost of:

$$C_1 = D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \tag{2.30}$$

while depth-wise, separable convolution has the computational cost of:

$$C_2 = D_K D_K \cdot M \cdot D_F \cdot D_F \tag{2.31}$$

The computational cost of separable convolution is:

$$C_3 = D_K D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_f \tag{2.32}$$

Where M denotes the number of input channels, $D_K \times D_K$ is the kernel size, $D_F \times D_F$ is the feature map size and N is the number of output channels. The two-step process of separable depth-wise convolution will lead to a reduction of computational cost of:

$$\frac{C_3}{C_1} = \frac{D_K D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_f}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2} \tag{2.33}$$

This can lead to an 8 to 9 times faster convolution with a small reduction in accuracy (Howard et al., 2017).

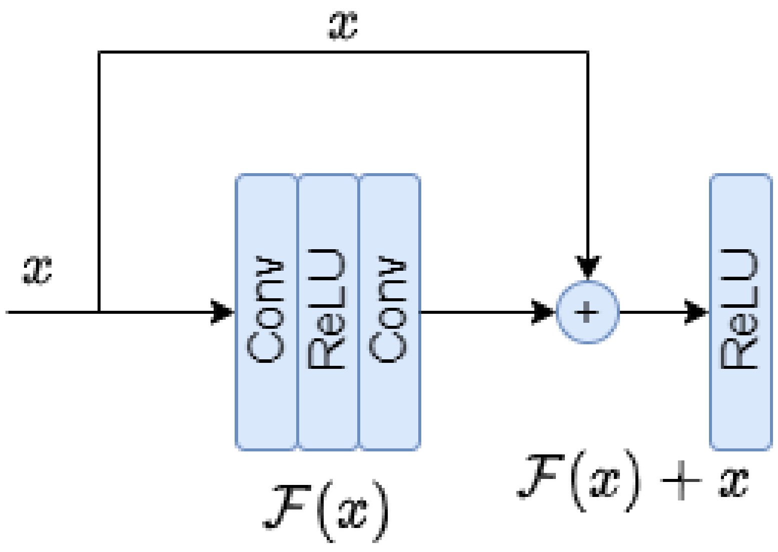


Figure 2.21: Residual block

Residual learning has also been popular to use in models, and in He et al. (2016b), a model of the ResNet model is proposed, utilizing residual learning. ResNet is based on the philosophy of VGG nets, with shortcut connections added to the model. ResNets learns residual representation functions instead of learning representation directly. Figure 2.21 shows how a residual block is structured with shortcut connections, or skip connections. The use of residual blocks will let the nonlinear layers fit the residual mapping $\mathcal{F}(x) = \mathcal{H}(x) - x$ instead of the underlying mapping, $\mathcal{H}(x)$, directly. Skip connections do not add extra parameters nor increase the computational complexity. The idea of ResNets is to solve the degradation problem that happens with standard models. The degradation problem is when a network depth is increasing, the accuracy gets saturated and decreases rapidly. Adding more layers will lead to higher training error. The idea behind residual mapping is that it is easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers with an optimal identity mapping (He et al., 2016b). Because of the skip-connections and the residual learning, the degradation problem is not present in the same way as with other models, such as VGG16. The ResNet can therefore contain more layers without the problem of a poorer accuracy. The baseline ResNet model have 34 weighted layers with fewer filters and complexity than the VGG nets.

Segmentation Models

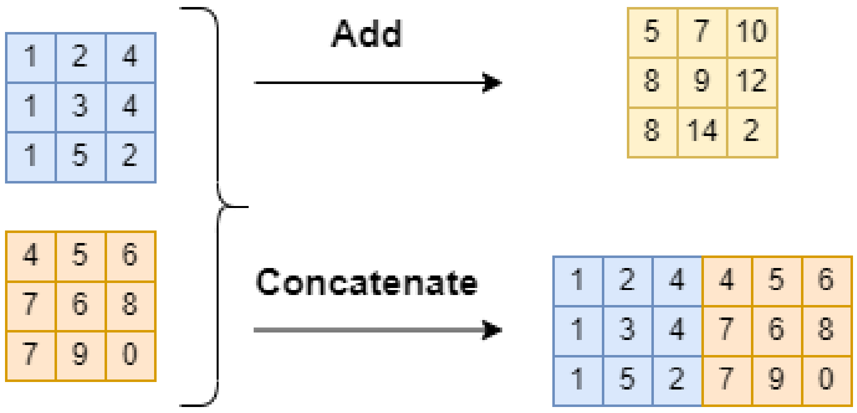


Figure 2.22: Skip-connection added and concatenated

U-Net (Ronneberger et al., 2015) is a model proposed for biomedical image segmentation. The network consists of a contracting part to capture context and a symmetric expanding path for precise localization. The U-Net is shown in Figure 2.10, where one can see that the model have a U-like symmetric structure. where The U-net utilizes skip-connections, such as ResNet, but instead of adding the shortcut-connection to the main path, the shortcut-connection is concatenated to the main path. The differences are shown in Figure 2.22. U-Net outperformed the SOTA methods in 2015, when the U-Net model was proposed and U-Net can be trained on very few images and achieve good results (Ronneberger et al., 2015)

PSP-Net (Zhao et al., 2017) is a model exploiting the global scene category clues with a pyramid pooling module. PSP-Net outperformed all other SOTA models in 2016 on the Cityscapes dataset. The PSP-Net uses a pretrained ResNet for extracting the feature map, which is of the size $\frac{1}{8}$ of the input image. A pyramid pooling module is used on top of the feature map as a global contextual prior. The pyramid pooling module pools the feature maps to new subregions at different scales. The bin-sizes are (1×1), (2×2), (3×3) and (6×6). Convolutions are done on the different subregions, up sampled and concatenated together with the original feature map. The different scale of bin-sizes will capture different levels of information, where the coarsest, (1×1) bin is a global pooling over the entire image.

The DeepLabv3+ model (Chen et al., 2017) utilize atrous convolution, a tool to adjust a filter's field-of-view to handle the problem of segmenting objects at different scales. By changing the rate of atrous convolution, the field of view will increase. With rate = 1, atrous convolution and standard convolution are equal. With rate = 2, a (3×3) atrous convolution kernel will convolve with every other element in a (5×5) area in the input during one convolution. The model contains modules with atrous convolution in cascade and parallel to capture multiscale information by adapting the atrous rates. Deeplab also contains a pyramid pooling mode, similar to the PSP-Net. DeepLabv3+ achieves better results than previous DeepLab-models and has, in 2017, and with comparable results to other SOTA-models (Chen et al., 2017).

2.3 Underwater Image Enhancement

Underwater images normally have a poorer visibility than images in air. The visibility underwater will be restricted to around twenty meters in clear water and only a few meters in turbid coastal waters. Both the specific propagation properties and transmission properties of light in water. The Beer-Lambert-Bouger law gives a relationship between the attenuation of water and the medium:

$$E(r) = E_0 e^{-ar} e^{-br} \tag{2.34}$$

Where E is the illumination of light, r is the distance, a is the absorption coefficient and b is the scattering coefficient of the water body. This law can be used as a basis for underwater enhancement methods (Xiong et al., 2020).

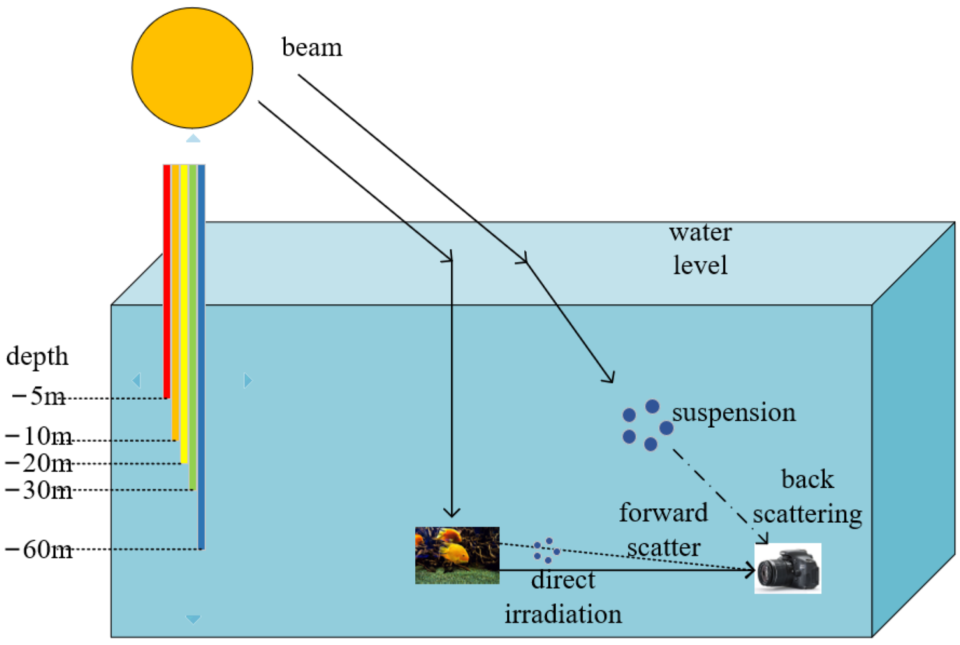


Figure 2.23: Underwater imaging model. Courtesy: Hu et al. (2022)

Figure 2.23 shows what can affect the quality of an underwater image. As the light goes through the water, it will be scattered and absorbed. Light with different wavelengths will propagate through the water differently. The light with the longest wavelengths will attenuate first, and below 60 meters, blue light will disappear. When a flash or camera light is used to capture images or videos, backscatter can occur, caused by small particles normally present in water. Forward scattering is also common in the underwater scene, where light from other sources randomly deviating, causing blurring of image features.

Vision enhancement methods can be used to remove some of the unwanted image degradation such as color bias, color loss, low contrast and atomization, light scattering, loss of intensity and blurring.

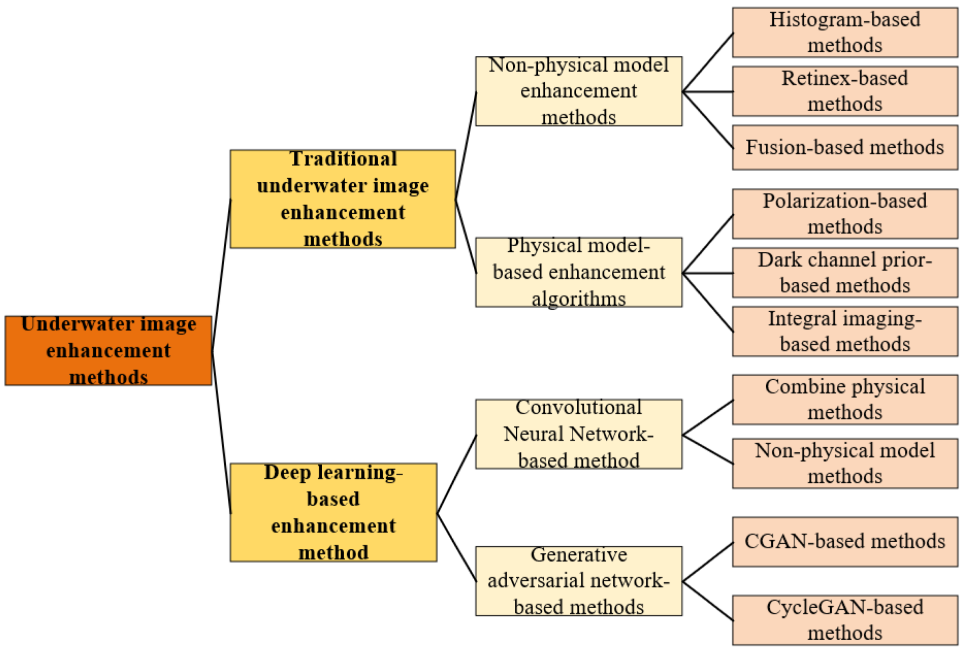

Underwater image enhancement methods
Traditional underwater image enhancement methods
Non-physical model enhancement methods
Histogram-based methods
Retinex-based methods
Fusion-based methods
Physical model-based enhancement algorithms
Polarization-based methods
Dark channel prior-based methods
Integral imaging-based methods
Deep learning-based enhancement method
Convolutional Neural Network-based method
Combine physical methods
Non-physical model methods
Generative adversarial network-based methods
CGAN-based methods
CycleGAN-based methods

Figure 2.24: Groups of underwater enhancement methods. Courtesy: Hu et al. (2022)

Several methods have been proposed to color correct and enhance underwater images, and can be divided into the categories shown in Figure 2.24. Contrast limited adaptive histogram equalization (CLAHE) (Yadav et al., 2014) is a non-physical model based on image statistics that uses histogram equalization to improve the contrast in the image. In Yadav et al. (2014), this technique is used to improve the visibility of foggy images. This can also be used underwater, because the light scattering underwater will act the same way as fog in air.

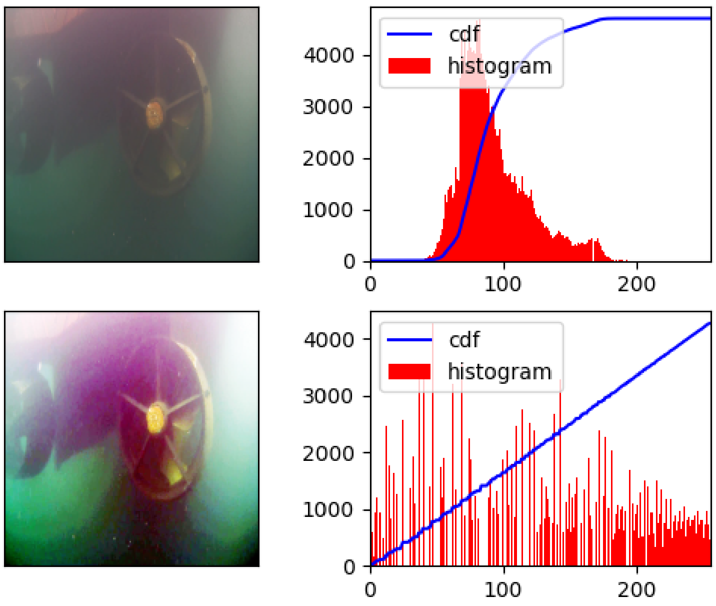


Figure 2.25: Normal image and histogram equalized image.

Figure 2.25 shows an underwater image where the color channels are histogram equalized by linearizing the cumulative distribution function (CDF) of the pixel values, as shown in the figure. The CDF can be defined as:

$$CDF_x(i) = \sum_{j=0}^{i} p_x(x = j) \tag{2.35}$$

where p_x is the image's pixel value. The equalized histogram can be found with:

$$g_{i,j} = floor((L-1)) \sum_{n=0}^{f_{i,j}} p_n \tag{2.36}$$

where $g_{i,j}$ is the new histogram equalized value and L is the maximum pixel value.

In adaptive histogram equalization (AHE), the enhancement process is applied over a specific region of the image and not on the complete image. Contrast limited adaptive histogram equalization (CLAHE), is a modified version of the adaptive histogram equalization and limits the contrast of the histogram. CLAHE was used in Yadav et al. (2014) for a real-time system and produced better results than AHE. In Vyavahare and Thool (2012), the images preprocessed with CLAHE gave a better segmentation results than original images.

The use of the dark channel prior is another way for single image haze removal. The dark channel prior can be used together with a haze imaging model to estimate the thickness of the haze and produce a haze-free image. A depth map can also be obtained as a product of estimating the haze (He et al., 2009).

$$J^{dark}(x) = min_{c \in \{r,g,b\}}(min_{y \in \Omega(x)} J^c(y))) \tag{2.37}$$

Where J^{dark} is the dark channel prior, J^c is the color channel and Ω is a patch centered at x. The transmission $\tilde{t}$ by:

$$\tilde{t}(x == 1 - min_c(min_{y \in \Omega(x)}(\frac{I^c(y}{A^c})) \tag{2.38}$$

where A is the atmospheric light. The scene radiance can be recovered with the equation:

$$J(x) = \frac{I(x) - A}{max(t(x), t_0) + A} \tag{2.39}$$

where the transmission $t(x)$, is restricted by a lower limit, t_0. This will preserve haze in very dense haze regions. The atmospheric light is estimated by selecting the 0.1% brightest pixels in the dark channel, and then the pixels with the highest intensity in the original image is chosen as the atmospheric light.

In Iqbal et al. (2007), an image enhancement algorithm is used based on slide stretching on the RGB and hue-saturation-intensity(HSI) color models. The contrast is increased by stretching the RGB color model and saturation and intensity stretching on the HSI color model. This algorithm address problems such as light absorption and scattering effects in the underwater environment.

$$P_o = (P_i - c)x\frac{b - c}{d - c} + a \tag{2.40}$$

Equation 2.40 is used to stretch the R-channel, G-channel, B-channel in the RGB color model and the S—and I- components in the HSV color model, where P_o is the normalized pixel, P_i is the input pixel, a is the minimum desired pixel value, b is the maximum desired value, c is the lowest pixel value present in the image and d is the highest pixel value present in the image. With Equation 2.40, the ratio between the pixel values is kept constant.

2.3.1 Generative Adversarial Networks

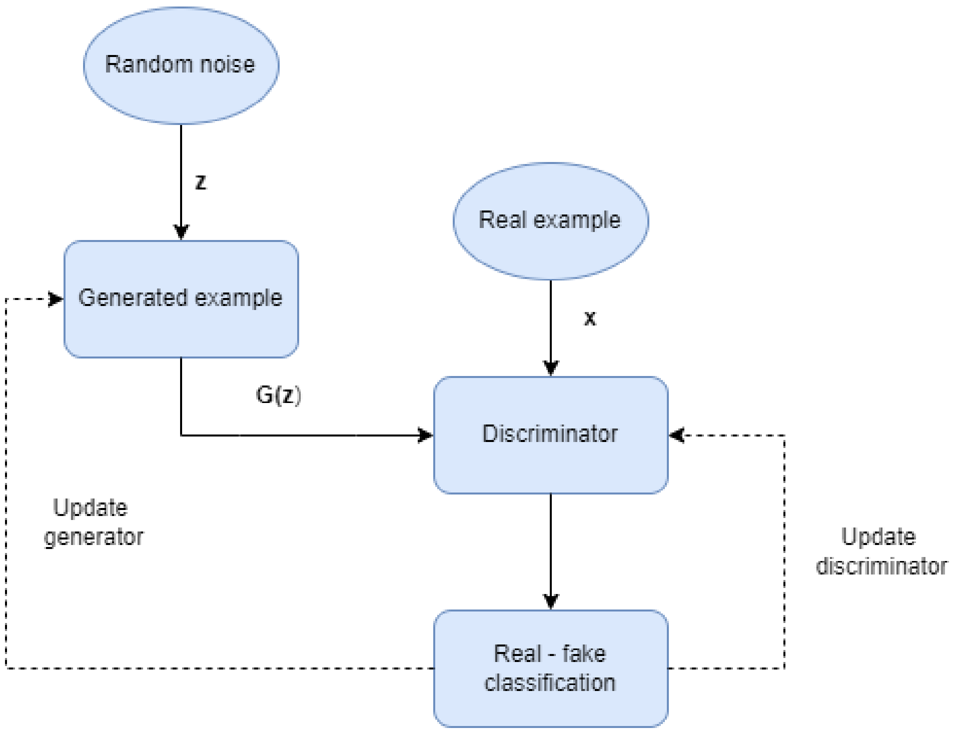


Figure 2.26: Simple architecture of a generative adversarial network

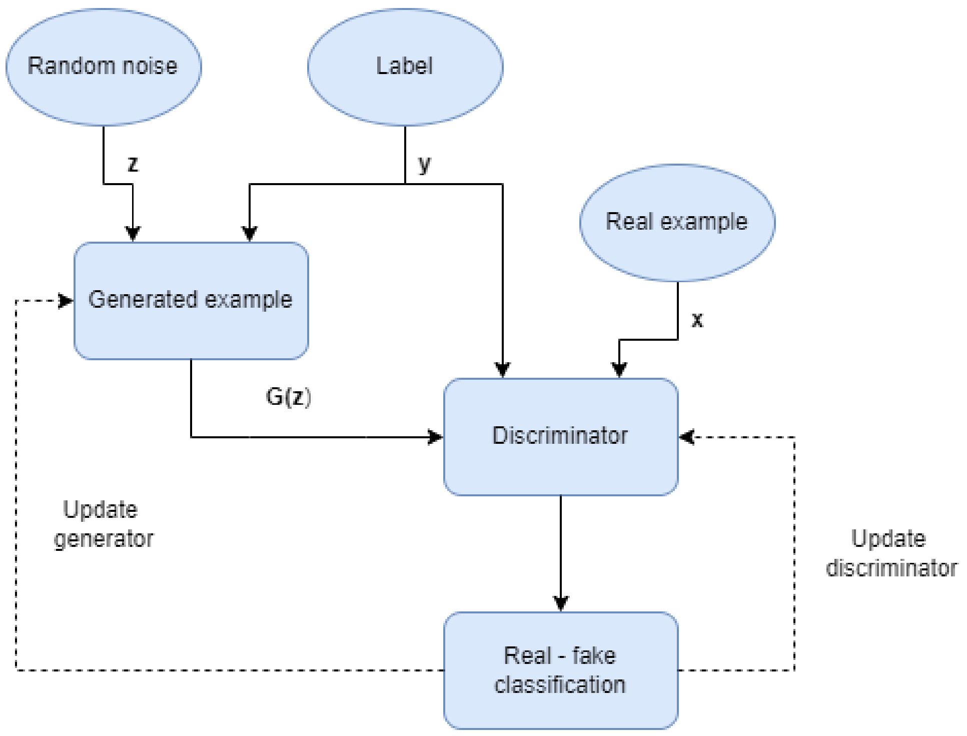


Figure 2.27: Simple architecture of a conditional generative adversarial network

Generative adversarial networks(GANs) are a relatively new type of neural networks proposed by Ian Goodfellow in 2014 (Goodfellow et al., 2014). The idea behind GANs is to estimate a generative model through an adversarial process. Two models are trained simultaneously; the generator and the discriminator, where the generator is trying to replicate the training data and the discriminator is estimating the probability that a sample came from the training data rather than from the generator. The objective for the generator is to maximize the probability for the discriminator to make a wrong estimation. The generator and the discriminator are set up in the generative adversarial network to play a two-player minimax game:

$$\min_{G} \max_{\theta_D} V(D,G) = \mathbb{E}_{x\ p_{\text{data}}(x)}[logD(x)] + \mathbb{E}_{z\ p_z(z)}[log(1 - D(G(z)))] \tag{2.41}$$

where G is the generator, D is the discriminator, x is the input data, $p_z(z)$ is noise that the generator is using to learn for variations in the mapping to the data space, $G(z;\theta_g$ where θ_g are the parameters in the generator represented by multilayered perceptrons. $D(x;\theta_d)$ outputs a scalar that represents the probability that x came from the input data. The first part of (2.41): $\mathbb{E}_{x\ p_{\text{data}}(x)}[logD(x)]$, represent the logarithmic probability of D predicting x to be real data, and the last part of (2.41): $\mathbb{E}_{z\ p_z(z)}[log(1 - D(G(z)))]$, represents the logarithmic probability of D predicting $G(z)$ to be generated data.

To avoid overfitting, the training is done by alternating between updating weights in the discriminator k times and then the weights in the generator k times. After the training is complete, with a successful training, the generator is able to generate similar data as the input data from a random input noise. Figure 2.26 shows a simple architecture of a GAN, where the discriminator tries to separate the generated data from the real data. The dotted lines represent the learning, where the GAN will alternate between updating the weights in the generator and the discriminator with interval length of k. The generator often have the same structure as fully convolutional neural nets, such as for example U-net.

The training of GANs can be seen as unsupervised learning. In unsupervised learning, the model trains on unlabeled data. The generator is provided self generated data to the discriminator, which can be seen as a supervised training supervised by the generator. The generator is training based on a loss function that the discriminator is learning during training.

A standard GAN uses a random noise as input to generate a new synthetic image. Another type of GAN is conditional generative adversarial networks (cGANs) (Mirza and Osindero, 2014). In a cGAN, the random noise is replaced by data, y, that is desired to be conditioned on both the generator and discriminator. The y values can represent low resolution images (Islam et al., 2020b), images of poor quality (Islam et al., 2020b), (Islam et al., 2020c) image style information (Gatys et al., 2016) or class labels (Isola et al., 2017). In a cGAN the generator uses y as input, while the discriminator uses both real / generated images, as with a GAN, and also the input data, y. A cGAN is still considered unsupervised learning, since the generator and discriminator is trained in the same unsupervised manner as the GAN, but annotated, or labeled, data is required to train a cGAN. In Figure 2.26 and Figure 2.27 one can see the differences between the two. The cGAN is based on conditional probability, where the minimax equation for GAN (2.41) can be rewritten with conditional probability for the cGAN:

$$\min_{G} \max_{\theta_D} V(D,G) = \mathbb{E}_{x\ p_{\text{data}}(x|c)}[logD(x,y)] + \mathbb{E}_{z\ p_z(z)}[log(1-D(G((z,y),y))] \tag{2.42}$$

where y is the conditional input.

The cGAN can be extended to a semi-supervised case by forcing the discriminator to also output labels that can be compared to the pre-annotated labels (Madani et al., 2018). This will give a similar result as conventional supervised learning, but instead of predicting N classes, the semi-supervised GAN will predict $N+1$ classes, where the last class is a “real-or-generated” class. This can be a more data efficient data-classifier and can also improve the generator’s ability to generate high quality samples.

Generative adversarial networks and conditional generative adversarial networks can have many use cases. In Li et al. (2021), a cGAN is used to create Synthetic images based on corresponding masks for a better generalization for a semantic segmentation task.Islam et al. (2020c) propose a GAN called FUnIE-GAN for fast underwater image enhancement for improved visual perception. FUnIE-GAN is both trained on paired data and unpaired data from the EUVP (Enhancing Underwater Visual Perception) dataset (The EUVP dataset, 2021). The EUVP dataset contains separate sets of paired and unpaired image samples with good and poor perceptual quality. The image enhanced with FUnIE-GAN achieves an increased performance on underwater object detection, human body-pose estimation and visual based attention-based saliency prediction of 11-14%, 22-28%, and 26-28%, respectively. Guo et al. (2020) proposes another GAN for underwater image enhancement with increased performance on key point matching and canny edge detection. Islam et al. (2020b) proposes a GAN called SESR for both image enhancement and super resolution and saliency prediction of underwater imagery. Both FUnIE-GAN and SESR achieve similar results when comparing the quantitative performance scores UIQM, SSIM and PSNR and outperforms other state of the art(SOTA) conventional image enhancement techniques such as RGHS, UCM, MS-Fusion.

2.4 Computer Vision

While machine learning and convolutional neural networks can perform well in object detection, conventional computer vision methods based on mathematical formulas is a robust technique for matching images, and can be used for 2D transformation of images, 3D reconstruction of scenes through matching of detected features in consecutive images and simultaneous localization and mapping (SLAM). This chapter will explain the concept of image matching by using a 2D transformation between two images.

2.4.1 Primitives and 2D Transforms

The primitives used here will be inspired by the primitives used in (Szeliski, 2021). 2D points in an image, or pixel coordinates, can be denoted with a pair of values: $\mathbf{X} = (x, y) \in \mathbb{R}^2$:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.43}$$

$\tilde{(\cdot)}$ will denote homogenous coordinates so $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathbb{P}^2$ will be a 2D point represented using homogeneous coordinates. $\mathbb{P}^2$ is called the projective space. A point represented by homogeneous coordinates can be converted back to inhomogeneous coordinates by dividing on the last element $\tilde{w}$:

$$\bar{\mathbf{x}} = \frac{\tilde{\mathbf{x}}}{\tilde{w}} = (\frac{\tilde{x}}{\tilde{w}}, \frac{\tilde{y}}{\tilde{w}}, 1) \tag{2.44}$$

Where $\bar{\mathbf{x}}$ is the augmented vector. IN inhomogeneous coordinates, $\tilde{w} = 0$ does not exist, as it represents points at infinity.

2D lines, **l**, can be represented with homogeneous coordinates as $\tilde{\mathbf{l}} = (a, b, c)$, which gives:

$$\tilde{\mathbf{x}} \cdot \tilde{\mathbf{l}} = ax + by + c = 0 \tag{2.45}$$

With homogeneous coordinates, the intersection between two lines can be computed:

$$\tilde{\mathbf{x}} = \tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2 \tag{2.46}$$

Where $\times$ is the cross product operator. Similarly, the line between two points can be found by:

$$\tilde{\mathbf{l}} = \tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2 \tag{2.47}$$

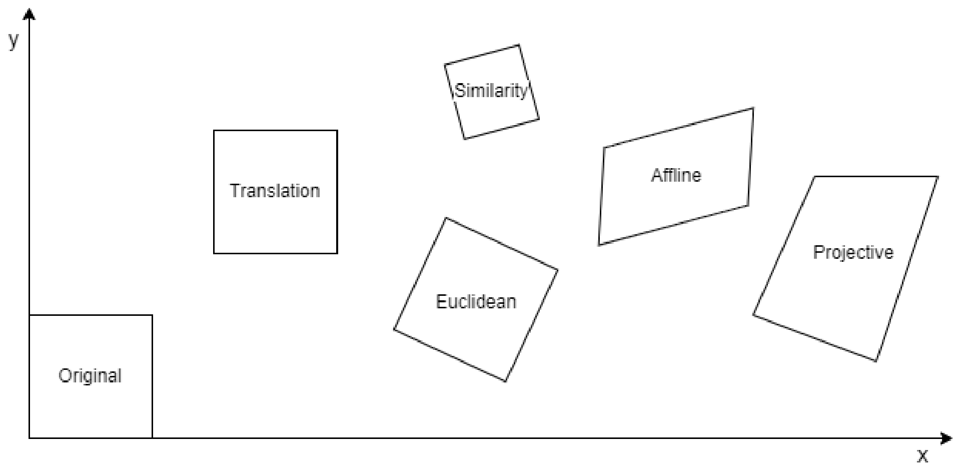


Figure 2.28: Different types of 2D transformations

There exists several types of 2D planar transformations, where translation, euclidean, similarity, affine and projective are examples of such transformations, shown in Figure 2.28. 2D translation can be written as : $\mathbf{x'} = \mathbf{x} + \mathbf{t}$:

$$\mathbf{x'} = \begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}} \tag{2.48}$$

or as:

$$\bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.49}$$

with **I** as the (2×2) identity matrix **t** as a (2×1) translation matrix and $(\cdot)'$ as a transformed value. Using Equation 2.48 gives a more compact equation with a (2×3) matrix, but Equation 2.49, which uses a full rank (3×3) matrix, makes it possible to chain transformations with matrix multiplications and to calculate inverse transforms.

Adding rotation to the translation transformation gives the 3D rigid body motion, or the 2D Euclidean transformation. It can be computed with the equation $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}} \tag{2.50}$$

Where **R** is a rotation matrix:

$$\mathbf{R} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{2.51}$$

where θ is the angle

When adding scaling to the Euclidean transformation, the transformation become a similarity transform:

$$\mathbf{x}' = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}} \tag{2.52}$$

The s in Equation 2.52 is a scalar scaling factor. The affine transform is a transformation where parallel lines remain parallel and can be written as $\mathbf{x}' = \mathbf{A}\bar{\mathbf{x}}$ where **A** is a (2×3) matrix.

The perspective transform, or homography uses homogeneous coordinates in the transformation equation:

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\tilde{\mathbf{x}} \tag{2.53}$$

Where $\tilde{\mathbf{H}}$ is a homogeneous (3×3) matrix:

$$\tilde{\mathbf{H}} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \tag{2.54}$$

$\tilde{\mathbf{H}}$ is only defined up to a scale. To obtain the inhomogeneous transformed coordinates, the homogeneous transformed needs to be normalized:

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}, \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}} \tag{2.55}$$

2.4.2 Feature Detection and Descriptor Matching

A perspective transform can be done if the transformation matrix, $\tilde{\mathbf{H}}$, is known. When matching two images with a perspective transform, $\tilde{\mathbf{H}}$ can be found by looking at corresponding coordinates in two adjacent images. To find corresponding image coordinates, feature detectors and descriptors can be used. Each detected feature has a corresponding descriptor that can be used to find the corresponding pixel location in another image. To create good match correspondences, the descriptors needs to be scale and rotation invariant. An example of a method for extracting distinctive image features is using the angle histogram describing the direction and magnitude of the image gradients around a keypoint. This method was presented in Lowe (2004). The image gradient can be found by using Sobels convolution kernel over the image to find the gradients, $\mathbf{G}_x$ and $\mathbf{G}_y$, in x and y direction:

$$F_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad F_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.56}$$

$$\mathbf{G}_x = \mathbf{I} \otimes \mathbf{F}_x, \quad \mathbf{G}_y = \mathbf{I} \otimes \mathbf{F}_y \tag{2.57}$$

Where the operator $\otimes$ is the discrete and bounded convolution. The magnitude of the gradient, $|\mathbf{G}|$, and direction of the gradients, θ can be found by:

$$|\mathbf{G}| = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}, \quad \theta = arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right) \tag{2.58}$$

Two common SOTA feature descriptors are Speeded-Up Robust Features (SURF) (Bay et al., 2008), Binary Robust invariant scalable keypoints (BRISK) (Leutenegger et al., 2011), and Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011). Surf is a scale- and rotation-invariant detector and descriptor. It uses integral images for faster convolutions. An integral image is the sum of all pixels at location $\mathbf{x} = (x, y)^T$ within a rectangular region around x:

$$I_{\sum}(\mathbf{x} = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} \mathbf{I}(i, j) \tag{2.59}$$

The detector in SURF is based on the Hessian matrix for good accuracy. The Hessian matrix, $\mathcal{H}$, for $\mathbf{x}$ at scale σ is denoted:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{2.60}$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the second order Gaussian derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$. The second order Gaussian derivatives are approximated by discretization and cropping. The use of integral images makes it possible to scale up the filters at constant cost instead of iteratively reducing the image size to make the features scale-invariant. The descriptor describes the intensity within the neighborhood of an interest point. The descriptor consist of a 4D descriptor vector $\mathbf{v}$ with the structure $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. $\mathbf{v}$ is calculated for a (4×4) grid around the interest point, making each descriptor containing 64 elements.

The ORB keypoint detector and descriptor was proposed as an efficient alternative to SIFT and SURF. ORB is a fast binary descriptor based on BRIEF (Calonder et al., 2010) and builds on the keypoint detector FAST. ORB is orientation invariant, resistant to noise and at two orders of magnitude faster than SIFT. ORB adds a fast and accurate orientation component to FAST, efficient computation of BRIEF features and a learning method for de-correlating BRIEF features under rotational invariance for better performance (Rublee et al., 2011). FAST are a good method for real-time keypoint finding, but FAST does not include an orientation operator such as for example SURF does. BRIEF is a feature descriptor using binary tests between pixels in an image patch. BRIEF is based on binary tests to train classification trees, which can be used to return distinct signatures for a keypoint once trained. ORB is based on a multiscale Harris keypoint and oriented patch descriptor that is used for image stitching. ORB adds multiscale features to FAST, by using FAST with a circular radius of 9 with a scale pyramid of the image with FAST features at each level of the pyramid. For choosing the N best keypoints, ORB deploys a Harris corner measure to order the FAST keypoints. To add orientation to the FAST features, ORB uses an intensity centroid to measure corner orientation. With Rosin's definition of the moments of a patch:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.61}$$

which gives the centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2.62}$$

Then, a vector from the center of the corner, O, to the centroid, C, can describe the direction of the patch:

$$\theta = atan2(m_{01}, m_{10}) \tag{2.63}$$

where tan2 is a quadrant aware version of atan. The BRIEF descriptor is an image patch made from a set of binary tests, τ:

$$\tau(\mathbf{p}; x, y) = \begin{cases} 1 & : \mathbf{p}(x) < \mathbf{p}(y), \\ 0 & : \mathbf{p}(x) \geq \mathbf{p}(y) \end{cases} \tag{2.64}$$

where $\mathbf{p}(x)$ is the pixel intensity at location x. The features can then be defined as a vector of n binary tests:

$$f : n(\mathbf{p} = \sum_{1 \leq i \leq n} s^{i-1} \tau(\mathbf{p}; x, y) \tag{2.65}$$

n is 256 in ORB. BRIEF is made rotation invariant by steering BRIEF according to the orientation of the keypoints. The steered version of BRIEF can be written as:

$$\mathbf{S}_\theta R \mathbf{R}_\theta \mathbf{S} \tag{2.66}$$

Where S is a (2×n) matrix with the location (x_i, y_i) of each binary test. $\mathbf{R}_\theta$ is the rotation matrix corresponding to the orientation θ of each keypoint. The resulting combination of the orientation invariant version of FAST and the rotation invariant version of BRIEF becomes ORB.

Matching the detected features and descriptors can be done in several ways. One technique is brute force matching, comparing the extracted features of one image against all features in another image and returning the best match. This is a slow method with a high computationally cost. Another method is FLANN (Fast Library for Approximate Nearest Neighbors). FLANN is a library that projects high-dimensional spaces to a lower dimensional space and generate compact binary code. With FLANN, nearest neighbor search algorithms such as kd-tree and locality sensitive hashing. Locality sensitive hashing groups point in space into buckets based on a distance metric. Points close to each other according to the distance metric are mapped to the same bucket. Kd-tree is a binary tree, where every node is a k-dimensional point. Every non-leaf node is splitting the k-dimensional space into two parts. Each new node will split one of the k dimensions into new half-spaces. After the tree is constructed from the k-dimensional features, the tree can be propagated to find the closest neighbor. The Kd-tree nearest neighbor search has an average time complexity of $\mathcal{O}(log(n)$.

After the Features are matched, two images can be stitched by using a projective transform. The homography matrix needed to do the projective transform can be estimated by the matched feature points $(\mathbf{x}_i, \mathbf{x}'_i)$. One method is least squares to minimize the sum of squared residuals,

$$E_{LS} = \sum_i ||\mathbf{r}||^2 = \sum_i ||\mathbf{f}(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i||^2 \tag{2.67}$$

Where $\mathbf{p}$ is the motion parameters that needs to be estimated and $\mathbf{f}$ is the transformation function between $\mathbf{x}$ and $\mathbf{x}'$. $\mathbf{r}_i$ is the residual between the measured location $\hat{\mathbf{x}}'_i$ and the predicted location $\tilde{\mathbf{x}}'_i$. Linear least squares is the simplest method for estimating the parameters, but the relationship between the unknown and the measurements are not linear in most cases. Normally, there also exist outliers, i.e., false matches that should not be incorporated in the estimation of the homography matrix. One widely used approach is RANdon SAmple Consensus (RANSAC). RANSAC selects a random subset of k correspondences to compute an initial estimate of **p**. Then the residuals of the full set is used to compute the residuals:

$$\mathbf{r}_i = \tilde{\mathbf{x}}_i(\mathbf{x}_i; \mathbf{p}) - \hat{\mathbf{x}}_i \tag{2.68}$$

Then, the number of inliers are counted. An inlier is classified as when the residual is smaller than a threshold, $||\mathbf{r}_i \leq \epsilon$. This is repeated S times and the sample with the largest numbers of inliers are chosen. These inliers are used to compute a final estimated homography matrix. Estimating the homography matrix, $\mathcal{H}$, from point correspondences with homogeneous coordinates:

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\Updownarrow$$

$$\begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i Y - i & y'_i \\ x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ -y'_i x_i & -y'_i y_i & -y'_i & x'_i x_i & x'_i y_i & x'_i & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.69}$$

Since the third row is a linear combination of the first and second row: row$_3$ = $-x'_i\cdot$ row$_1$ $-$ $y'_i\cdot$ row$_2$, every correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ gives two equations with eight unknown since $\mathcal{H}$ is homogeneous, so a minimum of four point correspondences are needed. To find non-trivial solutions to the equation $A\mathbf{h} = 0$ a single value decomposition can be used, $svd(A) = USV^T$ where the last column of V will correspond to **h**, which gives $\mathcal{H}$. $\mathcal{H}$ can be improved by using more than four corresponding through an optimization method such as Levenberg Marquardt by minimizing the reprojection errors. The Levenberg Marquardt method varies parameter updates between the Gauss-Newton update and the gradient descent update.

Chapter 3

Implementation and Method

3.1 Dataset

The dataset used in this project is a large-scale dataset for underwater ship Lifecycle Inspection, Analysis and Condition Information (LIACI) (Elvesæter, 2022). This dataset was presented in the paper "Semantic Segmentation in Underwater Ship Inspections: Benchmark and Dataset" (Waszak et al., In Press). The dataset consists of 1893 images with corresponding pixel annotations for 10 object categories. The Object categories classified in the dataset are: anode, bilge keel, corrosion, defect, marine growth, over board valves, paint peel, propeller, saliency, sea chest grating and ship hull. The paper proposes U-Net with MobileNetV2 as the encoder for semantic segmentation due to a balanced tradeoff between performance and computational efficiency.

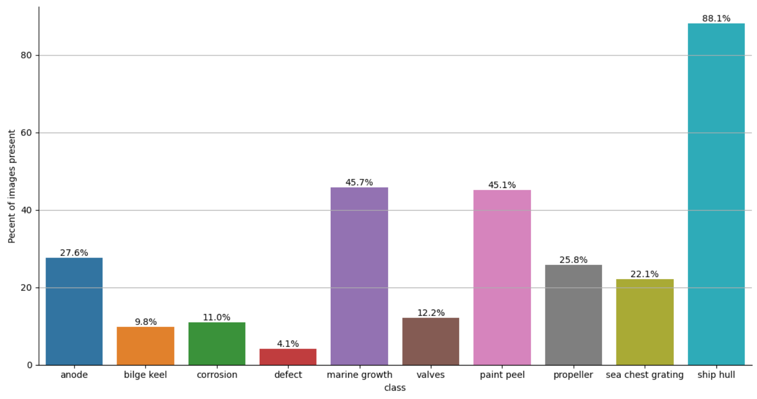


Figure 3.1: Percent of the images with each class percent

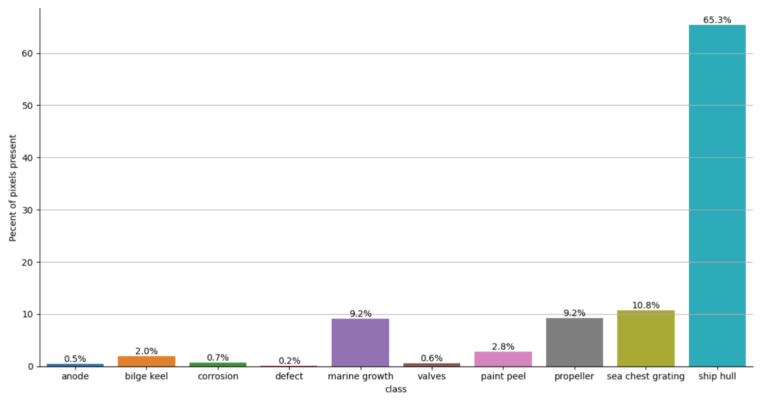


Figure 3.2: Percent of the pixels with each class percent

Figure 3.1 shows the distribution per image of the classes. The ship hull is present in the most images, with a presence in 88.1%. The defects have the lowest presence and are present in 4.1% of the images. When also looking at Figure 3.2, one can see that the size of the classes will vary. For example, are there anodes in 27.6% of the images, but the anodes are relatively small, so they will only be in 0.5% of the pixels.

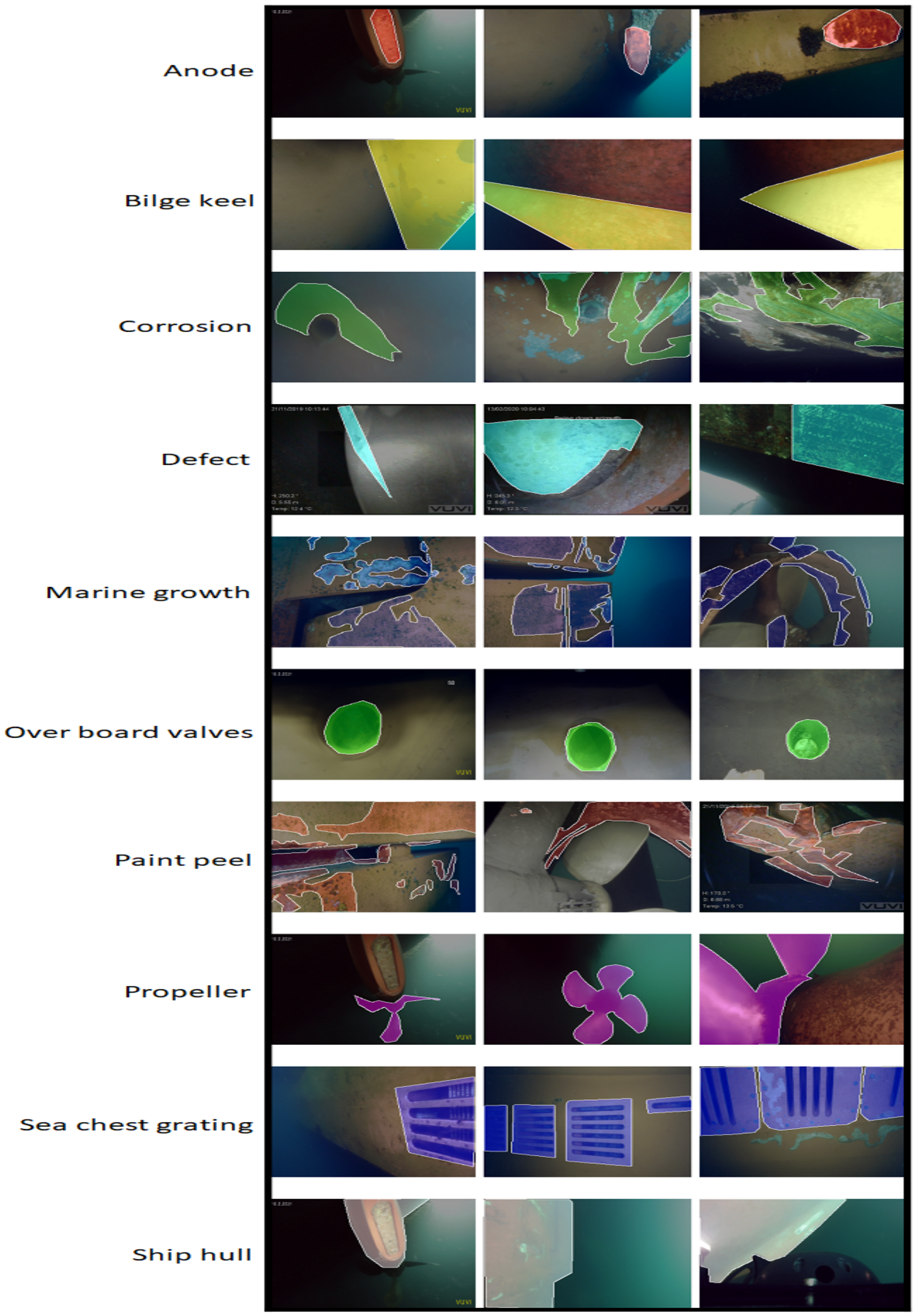


Figure 3.3: The labels in the dataset marked on images from the dataset

In Figure 3.3, one can see the different classes placed on images from the LIACI dataset. The data was labeled by two annotators and the annotations were quality checked by a professional ship inspector.

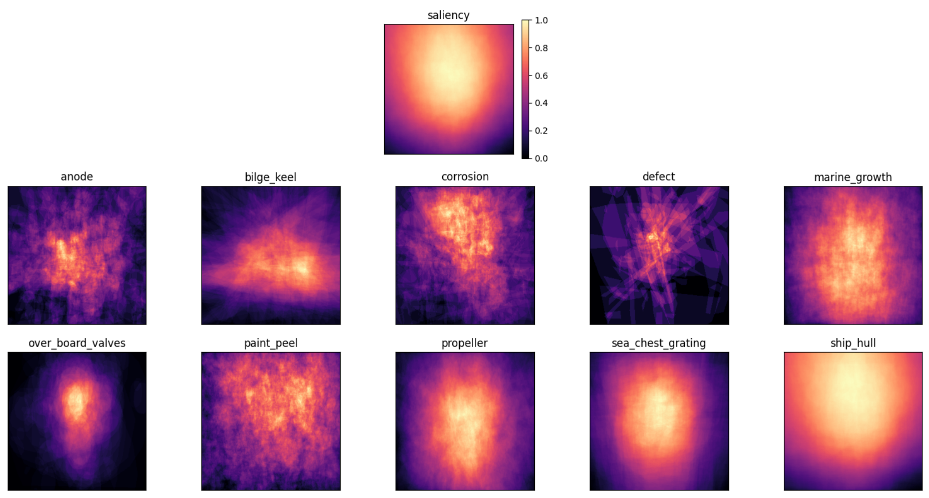


Figure 3.4: Average placement of classes

Figure 3.4 shows where the average position of the pixel annotations for each class, normalized between zero and one. This information has been used so to some degree to determine how to augment the images.

One aspect that will make it easier to do semantic segmentation predictions on datasets such as the SUIM dataset, is that all pixels will have exactly one label, no pixels without a class and no overlapping labels. This means models trained on the SUIM dataset can post-process the predictions from the softmax output layer by using argmax on the output and setting the value to one and the rest to zero. In the post-processing on the LIACI dataset, the output is set to one if the value after softmax i above 0.5 and set to zero else. The predictions from the SUIM dataset can be more accurate, since the predictions will be determined by the relative predicted probability, while predictions from the LIACI dataset will be based on absolute predicted probability.

3.2 Framework

The neural networks in this project are implemented by using Keras (Keras, 2022) and TensorFlow (TensorFlow, 2022). TensorFlow is an end-to-end open source machine learning platform in Python used to execute low-level tensor operations on GPUs, CPUs and tensor processing units. Keras is a high-level deep learning API running on top of TensorFlow. Keras was developed for enabling fast deployment of ANNs without loosing performance or scalability. In Keras, ANNs can be built from scratch, or well known network structures can be loaded and alternated for custom use cases. Deep neural is built layer by layer with the possibility to connect layers across the model.Other frameworks that could have been used is PyTorch (PyTorch, 2022) which is comparable to TensorFlow.

For image processing, the open source computer vision library OpenCV OpenCV, 2022 is used. OpenCV has functionality for basic tools such as reading images from file, resizing images, going from RGB to grayscale concatenating images, and reading and creating videos frame by frame. OpenCV is also used to do 2D transformations with functionality such as feature detection, feature description and feature matching with state-of-the-art methods.

The training of neural network demands relatively much processing power, and training with GPUs is desired to speed up the training process. Light-weight training and testing is done on a laptop with an AMD Ryzen 9 5900HX CPU and the main training is done through PaperSpace Gradient (PaperSpace, 2022) and Google Colab (Google Colab, 2022), which offers cloud computing with GPUs such as Nvidia Quadro RTX 5000 available.

Another computer vision tool used is Albumentations (Buslaev et al., 2020). Albumentations is a fast Python library for image augmentations. The library contains a variety of image transform operations optimized for performance. In this project, Albumentations are used for image – and mask augmentation during the training to avoid overfitting Albumentations have good functionality for applying randomized image augmentation to both the images and the masks when required. All geometric changes done on an image must also be done on the corresponding masks, while all color changes may only be done on the images.

The HDF5 (HDF5, 2022) library and file format is used to store and process the dataset. The HDF5 file format is build for fast I/O (input/output) processing and storage. The HDF5 storage is especially efficient when using cloud computing since it is simpler to handle, and the I/O process is faster compared to a conventional directory setup.

The segmentation models U-Net and PsP-Net are implemented using the python library Segmentation Models (Yakubovskiy, 2019), an API containing four model architectures and 25 pretrained backbones.

3.3 Pre-Processing

The preprocessing is an important part of the process of training the deep neural networks for semantic segmentation or other applications. A good pre-processing of the data will be beneficial for the efficiency, performance flexibility and simplicity. The pre-processing is done before the training starts and also during the training.

The LIACI dataset, with the 1893 images and ten corresponding image masks, has a total size of 4.05 GB. The images are stored in PNG (Portable Network Graphics) format, while the image masks are stored as BMP (bitmap) files. Both of these file formats are lossless, which is important in machine learning such that each pixel keeps its true value. A data format like JPG (Joint Photographic Experts) will convert the binary image masks, matrices with values of zero or one, to image masks with pixel values close to zero and one, which is not favorable. The BMP images are converted to PNG images because the OpenCV library will read PNG files at a higher rate. The class “saliency” is removed from the dataset, since it will not be used in the machine learning model. The dataset contains images of size: (720×1280×3), (480×640×3), (1080×1920×3). The images are resized to the size (256×256×3). This size is chosen because the down sampling will not affect the performance with any significance (Pranoto et al., 2018). A square image shape is normally used for semantic segmentation and since $256 = 2^8$ the images can be down sampled by a factor of two, eight times, which is done in the fully connected neural networks implemented, without doing any rounding. The masks, which has a depth of 1, are also resized correspondingly. The total size of the dataset after the preprocessing is at 197 MB, which corresponds to a decrease of 95%.

The dataset is prepared for training by reading the files from the dataset directory and placing them in a HDF5 library, where the complete dataset is split randomly into one HDF5 file for training, one HDF5 file for validation and one HDF5 file for testing. The dataset increases in size to 1.5 GB when the HDF5 file format is used, but the computation time decrease outweighs the disadvantage of the size increase.

For the data preprocessing during training, a custom data generator is implemented, where the basis of the generator is based on code from Keras HDF5 ImageDataGenerator, 2020. There exists functionality in Keras for generators, but when the data is stored in HDF5 file formats, Keras did not have good enough possibilities to customize the data generation for specific task. The data generator is reading the images and masks in specified batches, normalizes the pixel values, applies random data augmentation with the Albumentations library, reshuffles the dataset after each epoch and reformats the data to the correct shape. The input to the neural network needs to have the shape: (batch size $\times$ pixel height $\times$ pixel width$\times$ input channels), while the output in a supervised semantic segmentation training have the shape: (batch size $\times$ pixel height $\times$ pixel width $\times$ output channels). In the data generator, several augmentations are used. The augmentations and the parameters are chosen to improve performance by reducing overfitting while keeping the essential information in the input. The same augmentations and parameters are used in all supervised trainings in this project to make the results more comparable. During the semi-supervised training of the generative adversarial network, fewer augmentation is used. The augmentations and parameters used for the supervised training are:

- **Horizontal flip**
  - Probability: 0.5
  - Flips the image and masks horizontally
- **Rotation**
  - Probability: 0.5, range $\in [-45°, 45°]$
  - Rotates the image a random angle within the range. The max rotation is kept at 45° because the spatial information regarding the vertical order of different classes, since the images will never be taken upside down with a ROV. For example, will the propeller be placed at the lower part of the hull. There will also rarely be annotated ship hull in the lower part of the image, as can be seen in Figure 3.4.
- **Rain**
  - Probability = 0.5, drop length = 2, drop width = 1, brightness coefficient = 0.9
  - Adds random rain to the image. The rain drops are set to small values to replicate particles in the water.
- **Gamma**
  - Probability = 0.5, gamma range $\in [80, 120]$
  - Changes the gamma with a gamma parameter randomly chosen from the gamma range.
- **Perspective transform**
  - Probability = 0.5, scale $\in [0.05, 0.2$
  - Performs a perspective transform on the image with random distances of the sub image's corners from the full image's corners chosen from scale.
- **Gaussian blur**
  - Probability = 0.5, blur limit$\in [1, 5]$
  - Blur the image with a Gaussian filter with kernel size chosen from blur limit. The filter size is always an odd number.
- **Sharpening**

  - Probability = 0.5
  - Sharpens the image with a sharpening convolution filter.

- **Color jitter**
  - Probability = 0.5
  - Randomly changes the brightness, contrast, saturation, and hue of an image. The parameters for the jittering are randomly chosen between zero and 0.2.

- **Tone curve**
  - Probability = 0.5
  - Randomly changes the relationship between bright and dark areas of the image.

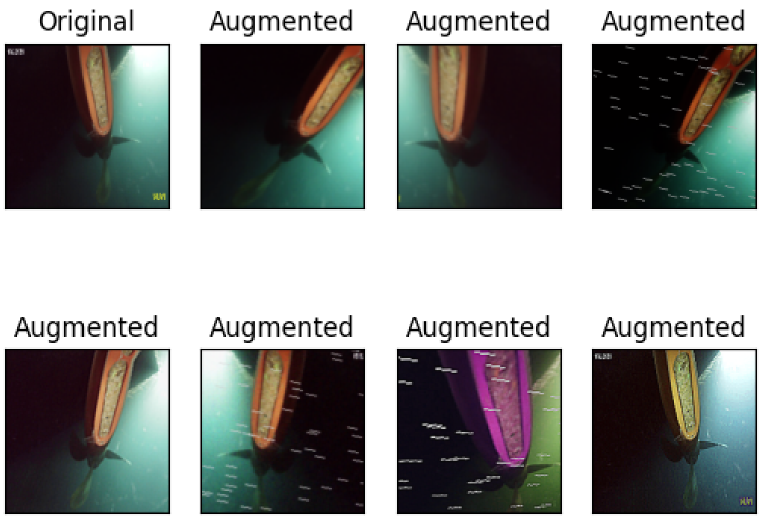


Figure 3.5: Image augmentation

In Figure 3.5, one can see the augmentation types and parameters chosen, applied on the same image seven times. The image will have different variations each time.

3.4 Segmentation Models

This section presents the selection of the models for semantic segmentation on the LIACI dataset and what considerations that are taken during the selection and changing of the models.

There exists many models that can achieve good results for semantic segmentation. For selecting a model, one approach is to train different models on the LIACi dataset and compare the results. The performance of different models will be compared, but the focus will be to use one model as a basis for testing different alternation to see how the performance is affected. Figure 2.19 is a good reference for choosing a base model, since the tests are done in a similar environment as the LIACI dataset, where the focus also is near real-time application. The importance of the similar environment-aspect is mentioned in section 2.2.4. In addition, models can vary in performance depending on the size of the dataset trained. As mentioned in section 2.2.5, U-Net will perform well when trained on a small dataset. The LIACI dataset and SIUM dataset contains approximately the same number of images (1893 and 1635, respectively) and a similar number of classes (ten and eight, respectively). In Islam et al. (2020a), the SUIM-Net models shows a high quantitative performance, as can be seen in Figure 2.19. For a real-time segmentation case, the prediction frame rate is important. The SUIM-Nets have the highest frame rates, where SUIM-Net$_{RSB}$ have the highest rate of 28.65 frames per second (FPS), while SUIM-Net$_{VGG}$ is the second fastest with 22.46 FPS (Islam et al., 2020a). SUIM-Net is therefore chosen as the base model in semantic segmentation training on the LIACI dataset. A frame rate of 22.46 FPS is more than high enough for real time segmentation, but this frame rate is measured on an Nvidia^{TM} GTX 1080 GPU. This is a powerful and fast processing unit, so the FPS will fall when the prediction is done on a GPU with less processing power or on a CPU, which is more probable to use in a real scenario.

3.4.1 SUIM-Net

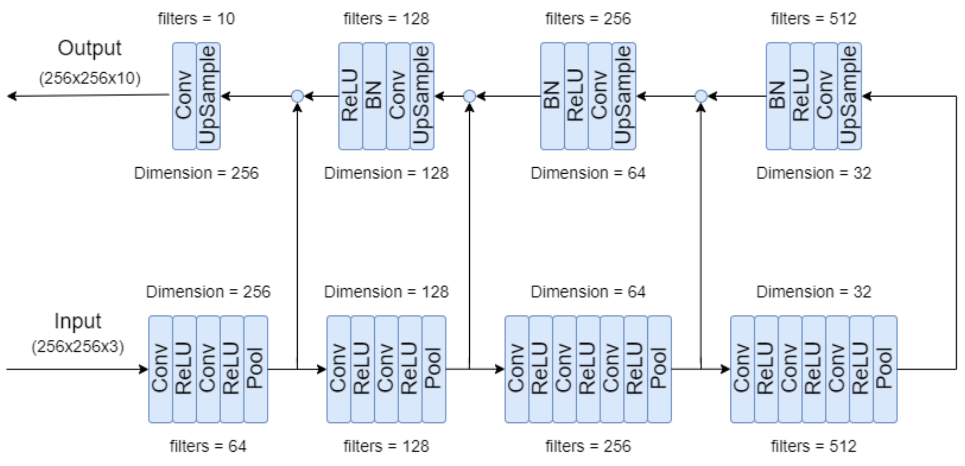
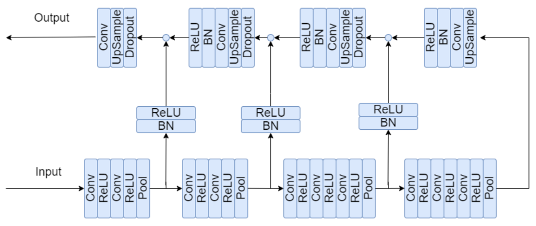
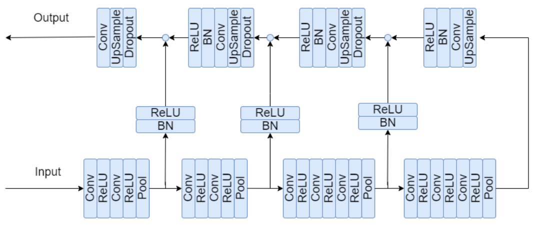


Figure 3.6: SUIM-Net

SUIM-Net$_{V}GG$ is based on a U-Net like structure, with a part of VGG-16 is used as a feature extractor. Figure 3.6 shows the structure of SUIM-Net$_{V}GG$. The encoder consists of the first four of the total five convolution-pool blocks in VGG-16, shown in Figure 2.20. A block have been defined as consecutive layers with no skip connections entering or branches out of the layers. The SUIM-Net uses skip connections right after the pooling layers instead of before, as done in a standard U-Net. This can cause loss of information, but can also contribute to increased generalizing capability during the training. The decoder blocks consists of an up-sample layer, one convolution layer batchnormalization and ReLU activation. The last block up-samples the filters to the same dimension as the labels, and the last convolution layer has the same number of filters as the number of label masks, which is ten. Each skip connection is concatenated with a decoder layers of the same dimension. The convolutions in the SUIM-Net uses (3×3) kernels with stride equal to one and same padding to keep the dimension of the feature maps equal as before the convolution. The SUIM-Net contains 12.2 million parameters, which is a relatively small network. By using transfer-learning with weights from the Imagenet dataset for the VGG16-encoder, the training will go faster than a large network trained from scratch. The model for the SUIM-Net$_{VGG}$ implemented in Keras is gotten from $https://github.com/xahidbuffon/SUIM$.



(a) SUIM-Net with batchnormalization



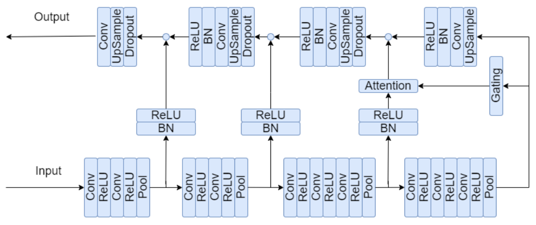
(b) SUIM-Net with batchnormalization and dropout

Different versions of the SUIM-Net is implemented to see if alternations can improve the quality of the predictions. It is desired to keep the transfer learning from Imagenet, so the encoder is not alternated. Three different versions of SUIM-Net are implemented and compared in a five-fold cross-validation test.

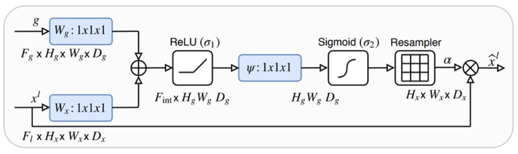
In section 2.2.2, batchnormalization and dropout are mentioned as methods to reduce overfitting and can help increase the generalization capability of the model. Results from Ioffe and Szegedy (2015) shows that it is better to use the non-linear activation layers after the batch-normalization layer, so ReLU is moved from directly after the convolution layer to directly after the batchnormalization layer. In the three skip connections in the model, a batchnormalization layer with a tailing ReLU activation layer is added. The resulting model is shown in Figure 3.7a.

An extension of Figure 3.7a with spatial dropout is also implemented (Figure 3.7b). As mentioned in section 2.2.2, dropout together with batchnormalization can also be beneficial for the generalizing capabilities of the model. Spatial dropout is chosen since this have been shown to be more effective than other dropout variants (Spilsbury and Camps, 2019).

Attention



(a) SUIM-Net$_{att}$



(b) Attention gate. Courtesy: Oktay et al. (2018)

Figure 3.8: SUIM-Net with attention

The third version of SUIM-Net implemented, is an extension of Figure 3.7b, shown in Figure 3.8a. The extension is the concept of attention, where the idea is that the model will learn where to look for objects of interest. The attention gate produces a mask, which is multiplied with the original skip connection Ideally, the attention mask will have values close to one in relevant parts and values close to zero in background regions. This will cause the attention mask to actively suppress activations at irrelevant regions. The implemented attention gate uses code from Attention UNet, 2019 and modified to fit the SUIM-Net.

The attention gate model was proposed in Oktay et al. (2018), where the attention module is used in a medical imaging case to locate pancreas. The attention gate is placed in every skip-connection in a U-Net. The results in Oktay et al. (2018) shows that the U-Net with attention outperforms a standard U-Net with 3.34% with F_1-score as the quantitative metric. The approach proposed in Oktay et al. (2018) is modular, so it can be applied to other models for image analysis. The attention model was demonstrated to be beneficial for tissue/organ identification and localization. The attention gate module is shown in Figure 3.8b where x^l is the skip connection and g is the gating signal. The gating signal is coming from the previous layer and is gated through a convolution layer to match the desired dimension. g and x^l needs to have the same dimension to be added together. The gating signal is used in the attention gate because it comes from a deeper part of the network, and thus, contains a better feature representation. x^l comes from earlier layers and has a better spatial representation, as mentioned in section 2.2.1 and shown in Figure 2.6. The attention gate will try to combine the spatial and feature information in a better way than without the attention module. The attention coefficients $\alpha_i \in [0, 1]$ will prune the shortcut connection from $x^l$by multiplication: $\hat{x}_i^l = x_i^l \cdot \alpha_i^l$. It is suggested in Oktay et al. (2018), that for multiclass segmentation, the coefficient α can contain one coefficient for each class to learn to focus on a subset of target structures. In this project, the attention coefficient only contain one element per pixel to simplify by learning the salient region instead. The attention module uses additive attention, which is shown to be more accurate than multiplicative attention (Luong et al., 2015). The additive attention can be formulated as:

$$q_{att}^l = \psi^T(\sigma_1(W_w^T x_i^l + W_g^T g_i + b_g)) + b_{psi} \tag{3.1a}$$

$$\alpha_i^l = \sigma_2(q_{att}(x_i^l, g_i; \Theta_{att}) \tag{3.1b}$$

where σ_2 is the sigmoid activation function described in section 2.1.1. The set of parameters Θ_{att} contains the linear convolutions: W_x, W_g, ψ, and bias b_{psi}, b_g. The attention gate is only used on the deepest layer in the implemented model. This was decided by applying the attention gate at different locations in the SUIM-Net. Empiric results from Schlemper et al. (2019) shows that attention gates were less effective if applied to the earliest layers. In the original attention module, the number of filters in the gating signal is set to match the number of filters in the skip connection. In this implementation, the filters in the attention gate are set to match the number of filters from the attention gate. This was found to give a better performance through experimental testing

3.4.2 Other Models

In the implementation and testing, the SUIM-Net has been the base for different versions and different comparisons. To get an overview of how the SUIM-Net is performing compared to other SOTA-models on the LIACI dataset with the chosen setup and parameters, U-Net, PSP-Net and DeepLabv3 models described in section 2.2.4 are implemented. Because of the varying training time, it was chosen to only do the training on one training set, so a one-fold cross-validation will be done with these models. The tree models are trained with VGG16 as backbone for comparable results. The PSP-Net is built in a way that requires the input image dimensions to be divisible by 16 because of the pyramid pooling module, so the pre-processing generator was alternated to re-scale the images from (256×256×3) to (240×240×3) and the masks from (256×256×10) to (240×40×10). The U-Net and PSP-Net were implemented using Yakubovskiy (2019) and DeepLabv3 was implemented by using Keras. With DeepLabv3, the output from the last convolution layer in block three (I_1) and the last convolution block in block five (I_2) from VGG16 were chosen as outputs of the backbone. I_2 enters the atrous spatial pyramid pooling and I_1 used as the first layer in the decoder part of DeepLabv3+.

Table 3.1: Parameters in the implemented model

Models	Trainable parameters	Non-trainable parameters	Total parameters
SUIM-Net	12,226,378	1,792	12,228,170
SUIM-Net$_{att}$	15,706,059	5,248	15,711,307
U-Net	23,753,578	5,248	23,753,578
PSP-Net	10,045,770	5,120	10,050,890
DeepLabv3+	20,154,538	5,728	20,160,266

In Table 3.1, the parameters in the different models, where PSP-Net has the fewest parameters and U-Net has the most parameters. This means the U-Net is the largest network and probably with the slowest run time. The run time will also the complexity of the operations done in the models, so it cannot be determined if the U-Net is the slowest just based on the parameters. The non-trainable parameters comes from layers that are not altered during the training process, such as batchnormalization layers. The magnitude of non-trainable parameters is around 5000 in each layer, except in SUIM-Net because of the few batchnormalization layers.

3.5 Adversarial Models

When looking at Figure 3.1, one can see that the distribution of classes are uneven. It is especially clear when looking at the distribution at pixel level (Figure 3.2). It is natural that this occurs, since there are, for example, fewer defects on a ship than ship hull. This can be a challenge when training models to segment images based on so few examples. During training, the segmentation model will focus to learn the more prominent classes because it would be easier to minimize the cost function of a class with more labels. The model will have more information about this class available, making more accurate and more generalized predictions. One way to focus more on the low represented classes is to use a loss function that will penalize poor predictions more, so the model will be rewarded more by minimizing the classes with poor predictions than to minimize the good predicted classes further. This can be done with, for example, the focal loss function, described in section 2.2.2. Using loss functions for uneven classes can also reduce the performance, since the low represented classes can have not a good enough representation to be predicted well in the first place, and then the model will try to improve something that cannot be improved further.

More images can be added to the dataset for training with manually annotated labels, but it is time-consuming. The LIACI dataset was annotated twice and validated by an expert. It can also be challenging to acquire good quality and diversified images of the desired class.

Another method is to use generated images to augment the dataset. GANs have recently shown potential in assisting in training of supervised models (section 2.3.1. In this project, a cGAN is implemented and trained on the LIACI dataset to generate synthetic images based on the labels. This is done to test if it is possible to increase the performance of a segmentation model by augmenting the dataset with generated images. In this project, the existing labels have been used to create synthetic images. There are many possibilities for testing and the tests to be done is training with full training sets, generating new images based on the labels in the training set and adding the generated image and corresponding mask to the training set. This will create a dataset twice as large, with one real and one corresponding generated image based on the real image's label. This is to check if the generated images can add information which the real does not contain. The other test is to compare a model trained on the first half of a training set, to a model trained on the same half of the dataset, together with generated images based on the labels from the remaining half. This test will replicate the effect of augmenting a training with generated images based on new created labels.

The cGAN implemented is a pix2pix GAN, proposed in Isola et al. (2017), with code from Linder-Norén (2019) used as the basis of the model and adapted to fit this case. Alternations to the training and fine-tuning is done to get better results. One change added is to train the generator and the discriminator separately in intervals of more than one batch at the time. The pre-processing data generator has been specialized for this use-case, where fewer augmentations are added. The ten labels are combined into one RGB color image. Also, with the cGAN pix2pix, the inputs and outputs of are switched, so the segmentation label is the input and the image is used as the ground truth.

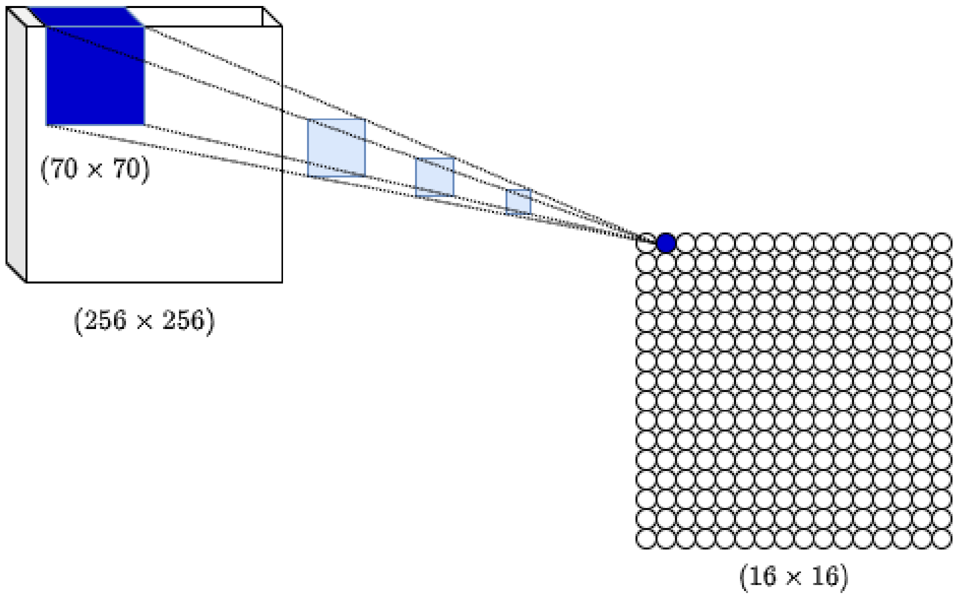


Figure 3.9: Concept of patchGAN

Table 3.2: Main layers in PatchGAN

Convolution layer	Patch	Kernel	Stride
1	(70×70)	(4×4)	(2×2)
2	(34×34)	(4×4)	(2×2)
3	(16×16)	(4×4)	(2×2)
4	(7×7)	(4×4)	(2×2)
5	(4×4)	(1×1)	(1×1)
6	(1×1)	-	-

The Pix2pix GAN implemented uses a U-Net as the generator and a discriminator based on a PatchGAN architecture. A PatchGAN evaluates patches of the image separately and uses the average of the individual predictions as the total evaluation. The discriminator is evaluating 256 ($16 \cdot 16$) different patches of the input image. The concept is shown in Figure 3.9. In Isola et al. (2017), patches of dimension (7×70) were proved to be more effective than PixelGAN (receptive field of(1×1)), smaller patches in PatchGAN (receptive field of (16×16)) and ImageGAN (receptive field of (256×256)). The discriminator was therefore altered to have a (70×70) receptive field. This was done by changing the last convolution from (4×4) kernels to (1×1) kernels. The PatchGAN has two inputs: the segmentation and an image, either real or generated by the generator. These inputs are concatenated channel wise to form a dimension of $(256 \times 256 \times 6$. In an additional implemented version of Pix2pix, the input to the generator is changed from a 3-channel input to a 10-channel input. In the ten-channel version, the dimension after concatenation will be (256×256×13). After the concatenation layer, there are five convolution layers, with batchnormalization after each convolution except the first. The kernels in the convolution layers are of size (4×4) with stride equal (1×1) or (2×2). The stride varies in size of the receptive field to be (70×70). Table 3.2 shows how the convolution layers are implemented to achieve patches of dimension (70×70) The receptive field at the previous layer before a convolution can be found by the equation:

$$RF = k + s \cdot (d - 1) \tag{3.2}$$

In (3.2), it is assumed in quadratic dimensions, where k is the kernel size, s is the stride, d is the dimension area where the receptive field is to be determined.

As activation function, leaky ReLU with $\alpha = 0.2$ is applied. The equation for leaky ReLU is:

$$f(x) = \begin{cases} x & : x \geq 0, \\ \alpha \cdot x & : x < 0 \end{cases} \tag{3.3}$$

With α = 0 in (3.3), leaky ReLU is equal to normal ReLU. Leaky ReLU is often used when sparse gradients can be a problem, such as in GANs. Leaky ReLU is also used as activation functions of the encoder in the U-Net generator.

In the results, the 3-channel version will be used in the tests. In the 3-channel cGAN, the discriminator is not altered, so the final sigmoid activation function is not present and the last convolution layer uses a (4×4) kernel, which gives the output a receptive field of (94×94) in the input image.

3.6 Image Enhancement

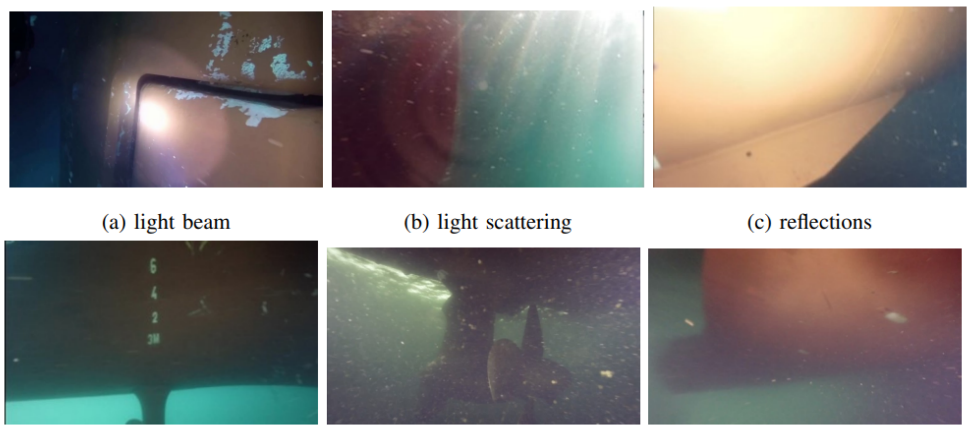

(a) light beam (b) light scattering (c) reflections
(d) scratches on lens protection (e) floating particles in water (f) water turbidity

Figure 3.10: Degrading effects on LIACI dataset. Courtesy: Waszak et al. (In Press)

Underwater images often suffer from image degrading effects, as looked at in section 2.3. Common underwater degrading effects are shown in Figure 3.10. Image enhancing have shown effects on improving the image quality, from quantitative metrics, from better feature matching and from a higher performance with CNNs trained on enhanced images. In this project, several image enhancement techniques will be implemented to see if it can improve a segmentation model's performance. The image enhancements will also be looked at in regard to the feature detection, since it has been shown to increase the feature matching.

The idea of enhancing as a part of the pre-processing is that the CNN-model can focus on learning other aspects of the image. A CNN can learn to enhance an image, if that will help the feature extraction and class localization. If image enhancement can replace a part of this task, the network could be reduced. This will optimally lead to that image enhancement combined with a model will give the same performance and faster run-time than a larger model. Another positive outcome is that image enhancement together with a model will give better performance than the model alone.

To test this, several enhancement techniques are implemented, where some of them are described in detail. For a diversified test, the enhancement methods chosen are based on different methods such as non-physical models, physical models and deep learning. The methods tested are:

- **Fusion based** The fusion based method is based on the paper Ancuti et al. (2012). This method enhances underwater images by fusing well known filters relevant for the underwater scene. Ancuti et al. (2012) shows improved segmentation performance, dehazing capabilities and improves image matching performance with SIFT operators. Model from Wang (2019).
- **Contrast limited adaptive contrast equalization** Described in section 2.3
- **Dark channel prior based**. Model from Wang (2019). Described in section 2.3
- **Integrated color model** Model from Wang (2019). Described in section 2.3
- Unsupervised color correction (UCM) Model from Wang (2019). This method is proposed in Iqbal et al. (2010) and is a method for underwater image enhancement. It is based on color balancing and contrast correction of both the RGB and HSI color model. The model removes the bluish color cast and improves low red color. The illumination and the true colors is also improved. The UCM method is shown to improve edge detection.
- **GAN based** The GAN-based enhancement methods tested in this project are not trained on LIACI dataset, but pretrained weights are used. The GANs are trained in similar underwater environments. The LIACi dataset is not suitable to train GANs for image enhancement, because it requires a dataset with two classes, consisting of degraded images and high quality images. FUnIE GAN was trained on EUVP and SESR trained on UFO-120
  - FUnIE-GAN Described in section 2.3.1. FUnIE-GAN is trained on the EUVP dataset, proposed in Islam et al. (2020c), consisting of paired and unpaired collection of 20,000 underwater images of good and poor quality. For the testing, the FUnIE-GAN weights used comes from training on the paired images. Model and model weights from Islam et al. (2020c)
  - SESR Described in section 2.3.1. SESR is trained on the UFO-120 dataset, proposed in Islam et al. (2020b), containing 1500 training samples. Model and weights from Islam et al. (2020b)

3.7 Video Segmentation

The application for the trained model in this project will be live video segmentation.

For testing the live video segmentation prediction, four videos from ship inspections were used. As an alternative, a physical experiment could have been performed with a ROV. Since the prediction model does not interact with the environment and only uses the video from the ROV, a video will give exactly the same output of information. Using a video for testing will also give the opportunity to redo the test with different models, parameters, or methods. The video is independent of the LIACI dataset and will give performance information from a new environment. The video replicates a physical test and will therefore not have a ground truth, i.e., no annotated masks are available in this test. Therefore, the results have to be mostly analyzed visually.

3.7.1 Visualizing Method

Table 3.3: Priority of the segmentation classes

Priority order	Class	RGB color code	Color
1	Marine growth	(0, 0, 127)	🟦
2	Corrosion	(0, 127, 0)	🟩
3	Paint peel	(127, 0, 0)	🟫
4	Defect	(0, 255, 255)	🟦
5	Sea chest grating	(0, 0, 255)	🟦
6	Over board valves	(0, 255, 0)	🟩
7	Anode	(255, 0, 0)	🟥
8	Propeller	(255, 0, 255)	🟪
9	Bilge keel	(255, 255, 0)	🟨
10	Ship hull	(255, 255, 255)	⬜
11	Background	(0, 0, 0)	⬛

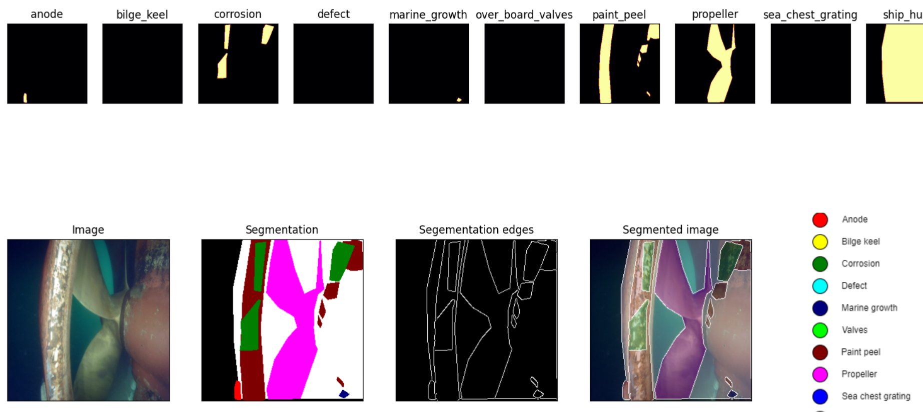


Figure 3.11: Visualization of the annotations

Figure 3.11 shows how the segmentation predictions can be visualized. The example is created from ground truth masks. The ten masks are merged together to a color image. Since the labels are over-lapping, an order have to be chosen for which labels to be prioritized when creating the three-channel segmentation image. The class priority is shown in Table 3.3. The edges are highlighted by using a canny edge detection on the segmentation. The canny edge detector is based on finding the intensity gradient of an image and suppress values based on a lower and upper threshold. The completed segmented image uses the canny edges to highlight the predictions. The image, segmentation, and segmentation edges are merged with to form the segmented image. The colors of the labels are chosen to be as different as possible with regard to the RGB value for making it easier for the GAN to separate the different classes.

3.7.2 Utilization of Image Sequences

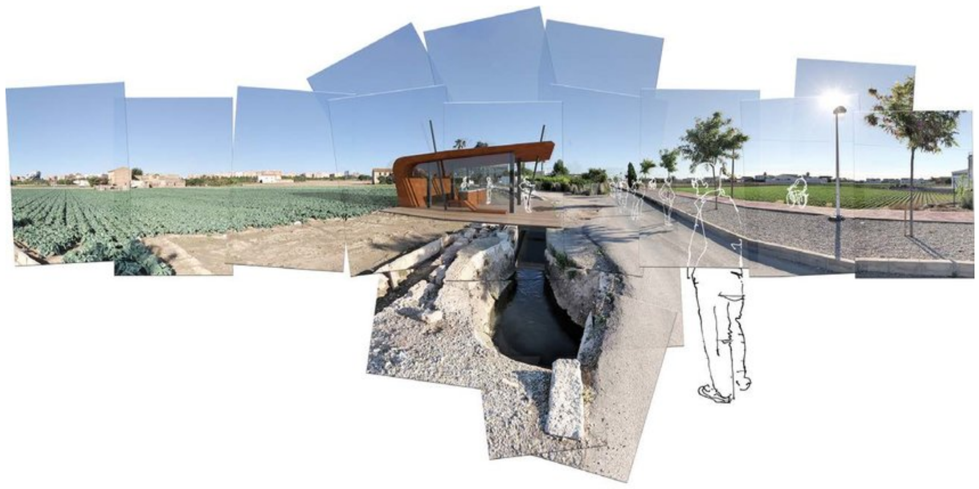


Figure 3.12: Panography. Courtesy: Costa and Iñarra Abad (2019)

Since the frames in a video will contain some of the same information as the previous evaluated frame, it is desired to utilize this for a better and less varying prediction. Several methods are considered

One method is to use image sequences during training. this is not possible to do with the LIACI dataset, since the images in the dataset are not in complete sequences. Some images are in a sequence, but with varying sequence length and with missing images in the sequence. If she image sequences were complete and of roughly the same length, R-CNN could have been used. An R-CNN can use the correlation from the last prediction in the current prediction, described in section 2.1.3.

Another method considered is to use other sensor information to determine how previous prediction information can be used. Sensors to measure velocity and distance would be relevant for this method.

The chosen method is to use a form of panography, shown in Figure 3.12. Panography, or image stitching, is using matched features between two images of the same scene from different perspectives, to combine the images. In the proposed method, the matched images are used to create a weighted average of current prediction and previous averaged predictions. Unlike in Figure 3.12, the segmentation predictions are transformed and combined, while the original images are used to find the correspondences. The theory behind the different steps in this method is described in section 2.4.

The method starts by a prediction of the first frame in the video. When the next frame evaluated, a new prediction is done. Then, ORB is used to extract 200 features and descriptors from each frame. Locality sensitive hashing is used together with kd-tree to find the two nearest neighbors to each feature in the previous frame. The two best matches are used to filter out uncertain matches with Lowe's test filter Lowe (2004), which states that if 75% of the distance to the second-best match is closer than the best match, the match will be disregarded, since it is not certain if the best match or the second-bast match is correct. The previous frame is used in the weighted average if the number of good matches is above a threshold.

The threshold is at seven matches. If the number of matches is below the threshold, the previous frame will be disregarded in the current prediction. If the minimum number of correspondences is reached, the homography is estimated with RANSAC. With the homography, the previous segmentation prediction is transformed to the current prediction. A weighted average is taken between the previous segmentation prediction and the current segmentation prediction where they are overlapping. The predicted segmentation is visualized as shown in Figure 3.11, but in the weighted averages, the ten masks are used with the continuous probabilities. When a new frame is evaluated, the weighted average prediction will be used as the previous prediction. It is used a weighted average instead of a standard average, since the transformations will vary in quality. The quality of the transformation is based on how much good data the homography can be based upon the number of good matches. Many good matches will imply an accurate overlap, and the previous prediction should be given more weight than the current prediction, since it already consists of several weighted prediction averages. If one object is present is five consecutive frames, and given six or more match correspondences between each frame, five predictions of the same object will be utilized in the current average prediction.

$$w_{linear} = \begin{cases} w_{min} & : m \leq m_{min}, \\ \frac{m_{max}-m}{m_{max}-m_{min}} \cdot (w_{min} - w_{max}) + w_{max} & : m_{min} < m < m_{max}, \\ w_{max} & : m \geq m_{max} \end{cases} \tag{3.4}$$

$$w_{cuadratic} = \begin{cases} w_{min} & : m \leq m_{min}, \\ (\frac{m_{max}-m}{m_{max}-m_{min}})^2 \cdot (w_{min} - w_{max}) + w_{max} & : m_{min} < m < m_{max}, \\ w_{max} & : m \geq m_{max} \end{cases} \tag{3.5}$$

(3.4) and (3.5) shows two methods to be used in the weighted average, Where w is the weight to be used in the weighted average, m_{min} is the lowest number of good matches, m_{max} is the limit where the weight reaches its minimum value, w_{min} is the weight at m_{min} and w_{max} is the weight at m_{max}. m_{min} is set to six, which is the minimum possible number of good matches, since the threshold already is set to five. m_{max} is set to 80. From observations, more than 80 matches will not increase the accuracy of the transformations. w_{min} is set to three, which means three times higher weighting to the current prediction than the previous, and w_{max} is set to 0.25, which means a four times higher weighing to the previous prediction than the current. The equation for the weighted is formulated as:

$$s_{avg} = \frac{w \cdot s_current + s_{previous}}{1 + w} \tag{3.6}$$

(3.4) shows a linear weighing function, which gives a linear relationship between increased matches and weight number. From observations, the accuracy of the transformations does not increase linearly with increased matches. A quadratic weight function is therefore proposed ((3.5)), which will match the accuracy of the transformation better. The accuracy will improve more when the number of matches increases from 15 to 20 than from 30 to 35.

The weighted average is desired to remove wrong predictions and give a more robust prediction when being able to predict from different perspectives. Since the predictions are originally completely independent frame by frame, the predictions can change fast and cause much variation, which is not desired. The average will work as a memory and reduce the variance. Additionally, the predictions are binary, i.e., the model is predicting true if the probability output is over 0.5 and 0 otherwise. With the average, the continuous probability can be utilized in a better way. As an example, if a pixel is predicted as to be marine growth with a probability of 0.95, 0.45 and 0.88 in three consecutive frames, the segmentation will be 1, 0, 1 while the average will be 1, 1, 1, which is probably correct.

3.7.3 Final Pipeline

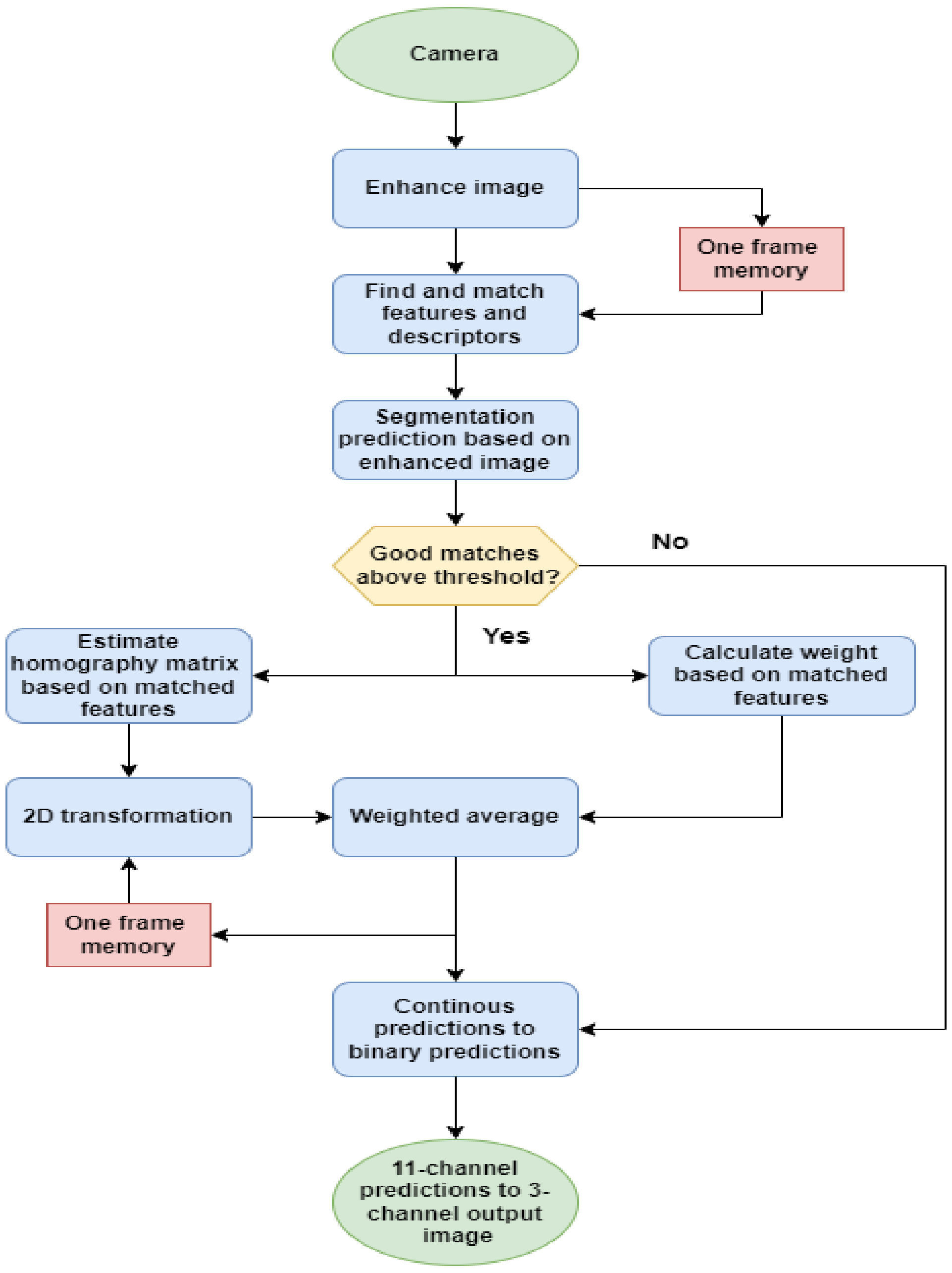

Camera
Enhance image
One frame memory
Find and match features and descriptors
Segmentation prediction based on enhanced image
Good matches above threshold?
No
Yes
Estimate homography matrix based on matched features
Calculate weight based on matched features
2D transformation
Weighted average
One frame memory
Continous predictions to binary predictions
11-channel predictions to 3-channel output image

Figure 3.13: Final pipeline for video segmentation

Figure 3.13 shows the final pipeline for video segmentation, where the methods implemented are combined to one model. The segmentation model predicts the probability for the presence of each class to be present present in each pixel, the image enhancement is used for better predictions and image matching and the computer vision algorithm is used for better predictions by combining information from previous frames.

3.8 Training

Training of the models is a large part of the project when considering the time aspect. The SUIM-Net uses 30 to 40 seconds per epoch in the training. When using transfer learning with encoder-weights from ImageNet, the model needs approximately 100 epochs for converging to the best performance on the validation set. Therefore, the number of parameters that can be changed and compared are very limited. A batch size of eight have been used in all training and the training is usually stopped when no improvement have happened in the last 40 epochs. The weights of the model as saved when loss is achieved on the validation and overrides the previous weights. The weights are saved to a H5 format, which is within the same category of files as the HDF5 file format. These weights are used for the performance tests, where the weights can be uploaded to new models with the same structure as during the training. A training set and a validation set are accessed by the data generator, batched, augmented and normalized. The training function is using the data generator to call for new batches. The data generator will alert the training function when one epoch has ended. When the epoch has ended, the order data is randomized. The learning rate used during training is 0.0001.

Chapter 4

Results and Analysis

In the tables, values marked with blue represents the best score and values marked with orange represents the second-best score. In general, the IoU score is used to measure the performance of the models. The IoU gives a good general quantitative measure of desired behavior. The IoU score is used as metric in percent, where the IoU is found with (2.29). Instead of calculating the IoU at each image for each class and using a form of average, the total number of true positives, false positives, and false negatives is found for each class and used to calculate the total IoU over the test set. The results consist of many comparisons since quantitative measurements is not always available available and improvements are more easily shown when compared to a baseline model. All predictions are also done test set, independent of the training data. The results are based on the LIACI dataset and four underwater ship inspection videos. Color codes are used when visualizing the segmentation maps, and the meaning of each color can be found in Table 3.3. The segmentation comparisons done in the results is either done with a five-fold cross-validation (Table 4.1 and Table 4.6) or a single cross-validation (Table 4.2, Table 4.3 and Table 4.5)

4.1 Model Comparison

Table 4.1: Five-fold cross validation of different versions of SUIM-Net. IoU is used as metric and scores are shown as $mean \pm \sqrt{variance}$

Models	**Anode**	**Bilge keel**	**Corrosion**	**Defect**	**Marine growth**	**Valves**	**Paint peel**	**Propeller**	**Sea chest grating**	**Ship hull**	**Average**
SUIM-Net	42.1±7.5	44.2±4.7	20.2±12.9	8.1±9.3	44.8±13.0	49.3±19.2	26.2±9.4	67.2±2.1	83.6±4.3	83.1±2.5	46.9±8.5
SUIM-Net$_{bn}$	42.2±3.9	42.9±2.6	17.9±12.7	1.0±1.0	50.7±2.5	51.1±16.3	27.9±2.5	67.8±3.0	85.4±3.4	85.4±1.0	47.2±4.9
SUIM-Net$_{bn+do}$	43.7±4.3	44.7±6.2	159±10.0	0.0±0.0	50.9±2.9	49.4±16.8	27.6±4.7	68.5±3.0	84.6±4.0	85.2±1.0	47.0±5.3
SUIM-Net$_{att}$	44.9±6.2	47.6±5.1	22.0±11.7	0.7±0.8	52.4±3.5	53.0±17.3	29.0±3.6	69.8±3.1	85.4±4.2	85.7±1.2	49.0±5.7

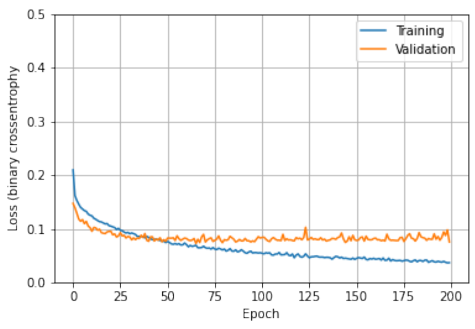
Table 4.1 shows the results of the five-fold cross validation comparison of the original SUIM-Net and three other versions. The alternations are done to test how these will affect the performance. Each net is trained and evaluated on five different randomly selected compositions of the LIACI dataset, where 80% of the data is placed in a training set, 10% is placed in a validation set and 10% is placed in a test set.

In Table 4.1, $(\cdot)_{bn}$ denotes added batchnormalization in the skip connection with ReLU moved to after the batchnormalization, $(\cdot)_{do}$ denotes added spatial dropout after the concatenations. $(\cdot)_{att}$ denotes attention gate implemented in the deepest skip connection. Dropout and batchnormalization is also used with the attention gate.

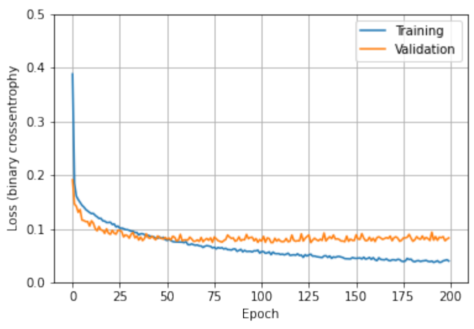
From Table 4.1, one can see that adding batchnormalization and dropout have little effect on the performance. This can be because batchnormalization is only added over the skip connections. Batchnormalization and dropout was not tested between the layers in the encoder because the pretrained Vgg16-encoder was used as a standard in all the tests to keep some consistency between the models and comparisons. When looking at the defect class, the batchnormalization will cause the model to fail almost all predictions with a IoU decrease of 80% to 100%. This can be seen as negatively, but the IoU is poor with the standard model too, and may not be considered as a good class to measure the performance. There are several reasons for this. A defect is a general term, and it can be difficult for a model to map from input to output in such classes. In Figure 3.3 one can see that the appearance of the defects can vary. In addition, the defect class is very underrepresented in the dataset and only present in 0.2% of the pixels (Figure 3.2). The defect class is also poorly distributed over the image (Figure 3.4). Because of this, the performance on the defect class will not be given much attention in the total evaluations. It is clear that the proposed model SUIM-Net$_{att}$ has the best performance. It has the highest score in eight of the ten classes and second-highest in one of ten. The average IoU score is 3.8% higher with the attention gate, which is a significant increase without adding too much complexity. The attention gate adds 3.48 million parameters to the original SUIM-Net, which is an increase of 28%. When comparing the prediction time of the two models, the increased run-time is only 5%. Therefore, the proposed model SUIM-Net$_{att}$ is the preferred model based on this cross validation. The standard deviation shown in $Table$ 4.1 shows that the models with batchnormalization normally have a lower value, which means a more consistent performance from these models.

Table 4.2: Comparison of different semantic segmentation models on the LIACI dataset. IoU is used as metric

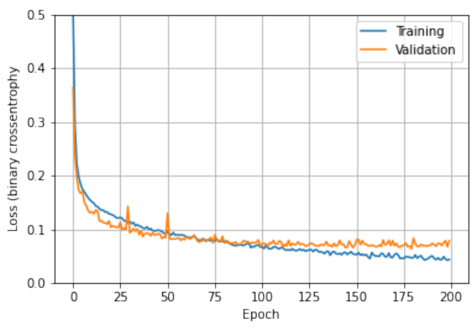
Models	**Anode**	**Bilge keel**	**Corrosion**	**Defect**	**Marine growth**	**Valves**	**Paint peel**	**Propeller**	**Sea chest grating**	**Ship hull**	**Average**
SUIM-Net	40.5	47.7	17.2	7.9	44.1	30.8	25.9	71.3	79.9	80.0	44.5
SUIM-Net$_{bn}$	45.2	44.2	13.4	1.8	50.8	31.6	24.5	71.5	84.8	86.4	45.4
SUIM-Net$_{bn+do}$	41.8	45.7	14.8	0.1	53.5	28.7	24.6	73.6	83.5	85.7	45.2
SUIM-Net$_{att}$	42.6	52.7	20.6	2.2	55.8	32.3	24.8	73.2	84.9	85.9	47.5
U-Net	25.3	63.9	9.0	0.0	51.6	33.1	21.2	81.3	85.2	88.5	45.9
PSP-Net	5.5	2.1	0.0	0.0	7.0	55.6	2.1	62.9	74.5	67.9	27.8
DeepLabv3+	20.3	52.3	9.3	0.0	54.0	31.9	9.4	74.8	87.5	87.9	42.7



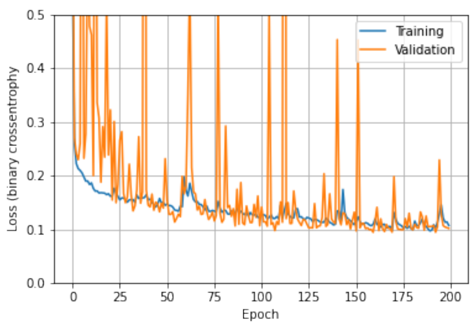
(a) SUIM-Net



(b) SUIM-Net$_{att}$



(c) U-Net



(d) DeepLabv3+

Figure 4.1: Training of different models

To get a better overview of how the SUM-Net variations are performing, they are compared to other SOTA models in Table 4.2. From Table 4.2, one can see that SUIM-Net$_{att}$ has the best average performance. DeepLabv3+ also achieve good results on the classes with high scores. U-Net have a good performance overall, close to the ferformace of SUIM-Net$_{att}$. PSP-Net does not achieve high scores. PSP-Net is the model with the fewest parameters, but a higher score was expected. In Figure 4.1, the binary cross entropy development is plotted during the training over 200 epochs. One can see that U-Net (Figure 4.1c) gets a lower binary cross entropy score than the SUIM-Net models (Figure 4.1a, Figure 4.1b) and has in general less overfitting problems. DeepLabv3+ (Figure 4.1d) have no overfitting, but converge towards a higher loss score than the other models. In Table 4.2, both U-Net and Deeplabv3+ were trained beyond 200 epochs until no improvement were occurring. Deeplabv3+ also have a high variance on the performance on the validation set during training, which can suggest some unstable training process.

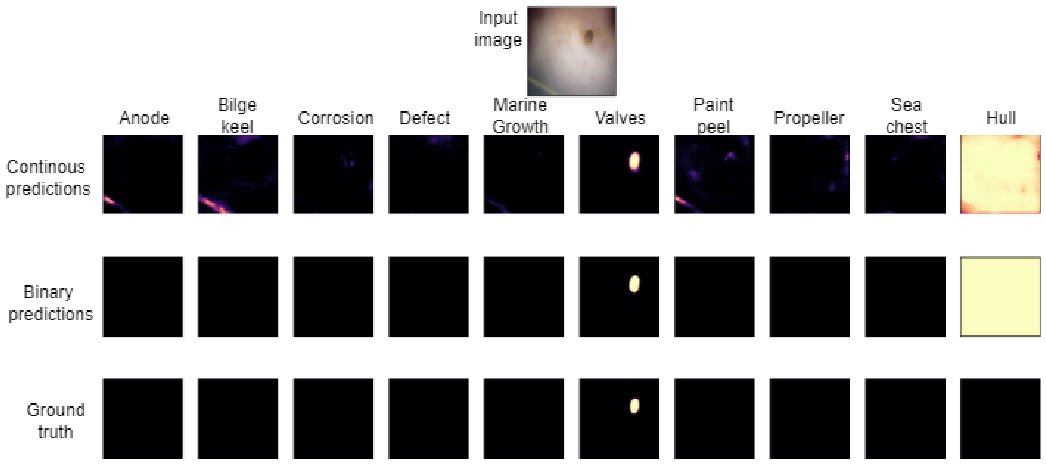


Figure 4.2: The continuous prediction and binary prediction. Example 1

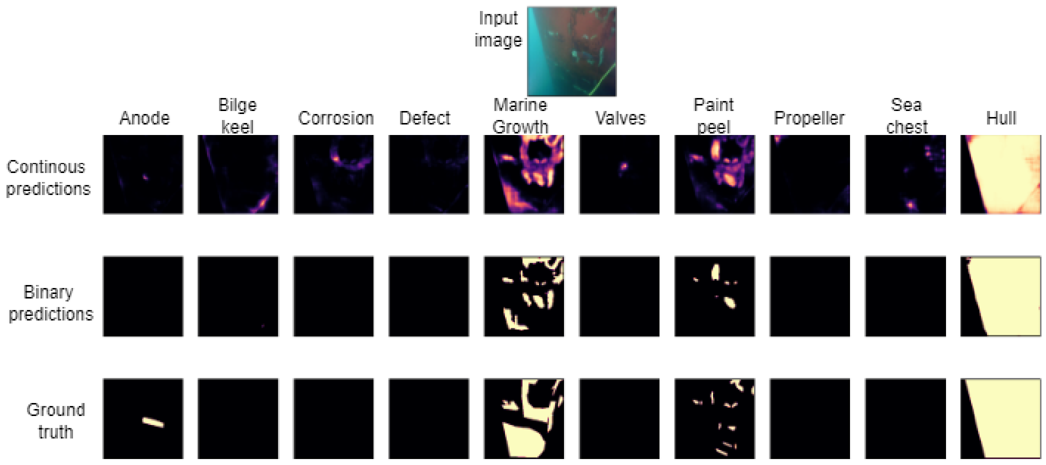


Figure 4.3: The continuous prediction and binary prediction. Example 2

Figure 4.2 and Figure 4.3 are two examples on how SUIM-Net$_{att}$ is doing predictions. The continuous predictions, which is a measure of how probable the model estimates that a class is present in the pixels, is transformed to binary final predictions. All the predictions that are estimated to be present, with probability of over 50%, are labled as positive predictions. The rest is marked as negative. One can see that the model can predict the presence of ship hull with a high probability, while classes such as marine growth are more uncertain. The model can understand that these classes are present in Figure 4.3, but cannot tell with a high probability exactly which pixels that the classes are present. Figure 4.2 can also predict the location of a valve with a high certainty.

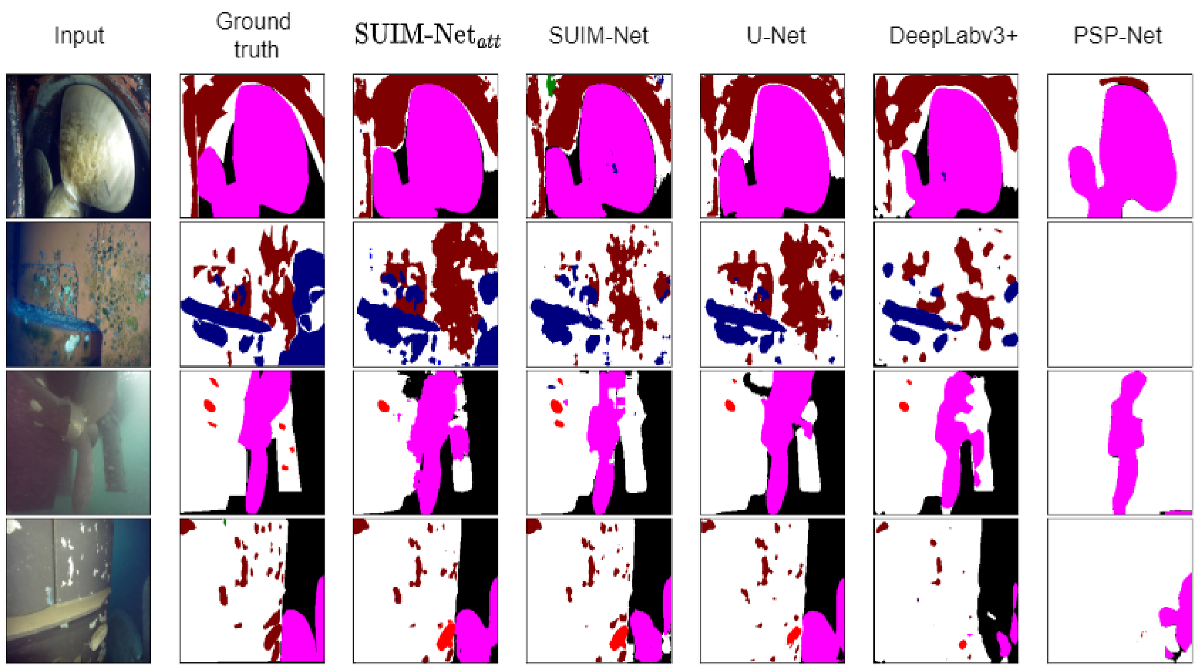


Figure 4.4: Segmentation prediction on four images with five models

Figure 4.4 is a visual representation of how the models are predicting. The four example images contains mostly the classes ship hull, propeller and paint peel, marine growth and anode. The predictions should, to a certain degree, be comparable to the IoU scores in Table 4.2. PSP-Net have the lowest IoU scores, and one can see in Figure 4.4 that it has the worst performance. It is predicting ship hull in the entire image, which can be understood by the IoU score for ship hull, where it got a score of 67.9%and since the dataset is containing ship hull in 65.3% of the pixels, it seems like the PSP-Net is predicting ship hull in all pixels. One can also see that the standard model have the most accurate predictions of paint peel, showed with maroon color, which coincides with the results from Table 4.2. It is difficult to compare the results visually, but from the author's point of view, U-Net predicts the propeller (pink) the best and SUIM-Net$_{att}$ predicts the marine growth the best, which coincides with the results numerical results and proves the validity of the results in Table 4.2 to a certain degree.

4.2 Image Enhancement

As described in section 3.6, several image enhancement methods with a focus on underwater images have been implemented. The methods are evaluated based on whether using enhanced images will increase the performance of a segmentation model and feature matching. Many of the methods implemented are evaluated, in their respective papers, by using quantitative measures estimating the image quality. The images have been evaluated visually to confirm the implementation is done correctly and that some improvements can be observed, but the visual improvements are not a goal in itself.

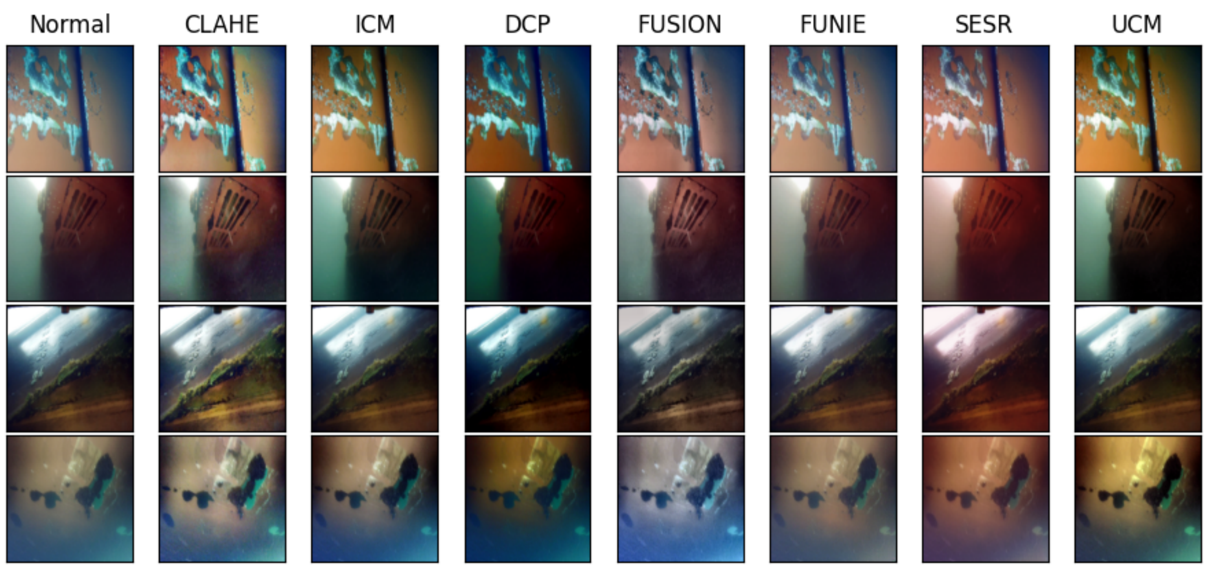


Figure 4.5: Comparison of seven different enhancement methods

In Figure 4.5, the enhancement methods are applied to four underwater images. One can see that the methods alters the images differently with different degree of change. Some images can also seem to be more degraded, as the fusion method at the bottom image, but the later test will determine the quality of the methods for this use case.

Table 4.3: Performance comparison of SUIM-Net trained together with different image enhancement methods

Enhancement	Anode	Bilge keel	Corrosion	Defect	Marine growth	Valves	Paint peel	Propeller	Sea chest grating	Ship hull	Average
CLAHE	41.3	54.4	24.9	3.8	49.2	32.3	22.9	71.0	83.9	84.6	46.8
ICM	42.5	47.2	31.8	4.9	45.3	30.9	29.5	70.0	82.5	83.0	45.8
DCP	48.4	54.9	14.3	0.5	48.7	22.8	21.4	68.8	83.2	82.4	44.5
FUSION	39.0	31.6	15.3	1.8	43.1	30.1	16.3	65.8	83.0	82.7	40.9
FUNIE	37.6	48.6	18.9	7.1	48.2	30.9	23.7	69.0	83.8	81.5	44.9
SESR	40.0	48.4	12.4	5.4	47.9	32.0	21.1	70.4	83.8	85.4	44.7
UCM	41.1	52.3	20.8	3.2	54.3	33.0	28.9	70.8	86.8	83.3	47.5
Normal	40.5	47.7	17.2	7.9	44.1	30.8	25.9	71.3	79.9	80.0	44.5

Table 4.3 shows a comparison where the standard SUIM-Net have been trained with enhancement methods applied to the images before the training. In Table 4.3, one can see that using the UCM and CLAHE method before training, the average IoU score is increased by 6.7% and 5.2%, respectively, which is a significant increase. The GAN-based method SESR performs poorer than the standard method. DCP and FUNIE gives a relatively equal average IoU than a normal image. ICM preprocessing also give an improved IoU score. The three methods that increases the performance of the model are based on one form of histogram stretching. They process the images in a pre-defined constant matter, which will make it possible for the segmentation model to learn a constant mapping between the input image and the segmentation classes. The idea behind using the other models was that they would remove variations that come from image degrading effects. Also, many of the images are taken close to the hull, where all water effects will not be present. In these images, all alternations based on a physical model would not cause any improvement and could lead to worse predictions. UCM shows the best average IoU-score and has the best score on several classes, but CLAHE has a very consistent good score compared to the other methods. It has the second-best average score and the second best IoU score in seven of ten classes. This makes CLAHE and UCM the preferred method from this test. Since it is a comparison based on one training, the results cannot be trusted completely, but give a good indicator of the performance.

Table 4.4: Average good matches between following frames in five videos

Enhancement method	Video 1	Video 2	Video 3	Video 4	Video 5	Average	Average relative difference
CLAHE	100.2	95.4	67.0	95.5	30.7	77.8	14.2%
ICM	77.9	99.2	68.0	92.6	34.7	74.5	9.6%
DCP	83.2	95.1	65.3	89.3	33.4	73.3	8.2%
FUNIE	59.1	95.0	64.2	89.1	25.2	66.5	-5.6%
SESR	64.0	95.3	63.6	87.9	30.6	68.3	-0.9%
UCM	80.5	100.3	68.8	93.8	44.6	77.6	17.5%
Normal	58.7	97.5	66.2	91.4	31.7	69.0	0%

The image enhancement methods can also increase the number of good features matched with the ORB feature detector and descriptor. In Table 4.4, the image enhancement methods are compared by reading five different underwater ship inspection videos frame by frame, run the implemented feature matching algorithm between each frame, described in section 3.7 and find the average good matched features. In Table 4.4, it is show that CLAHE has the highest average number of features matched. A new average is used in this comparison: the average of relative difference. This average will give a higher weight to improvements in the videos with low average matches. This metric is used because an increase from for example 30 to 35 matches is more important than an increase from 90 to 95 matches. The importance of this is described in section 3.7.2. The average of relative difference is done by using the feature matching on normal images as the baseline and take the average of the percentage increase of matches in each video. With the average relative difference, the UCM method has the best performance. The ICM and DCP method gives better feature matching, while the GAN based methods gives a worse performance than the baseline. The UCM have consistent good results in this test, being the only method with a higher number of matches than the baseline in all the videos.

From the two tests with enhancement methods, it is the CLAHE method and UCM method that shows the most potential, with a significant increase in performance in both cases tested. The GAN based methods gave a lowers score in both cases. One reason the GAN model performed poorly can be because they were not trained on this dataset. When comparing the run-time of CLAHE and UCM, CLAHE is the fastest with a runt-time of under 0.0005 seconds per image. In the initial implementation, UCM had a run-time of 1.2 seconds per image and was improved to 0.1 seconds per image. The run time can probably be improved more, but this part has not been a big focus area in this project. CLAHE will therefore be used as the proposed method for image enhancement for increasing the performance in semantic segmentation and feature matching because of the consistent increase in performance in segmentation and the negligible run-time.

4.3 Generated Images

In this section, the effect of using generated images to augment a dataset is tested. The implemented Pix2Pix GAN is trained on the dataset and the images are generated based on the already existing segmentation masks merged to a 3-channel color segmented image. Another version of the GAN is also implemented and trained that is based on the binary segmentation masks, where the GAN takes a 10-channel input. This is done so that the same labels used in the semantic segmentation can be used with the GAN also without any intermediate steps. The 10 color-channel GAN is not used in any of the tests.

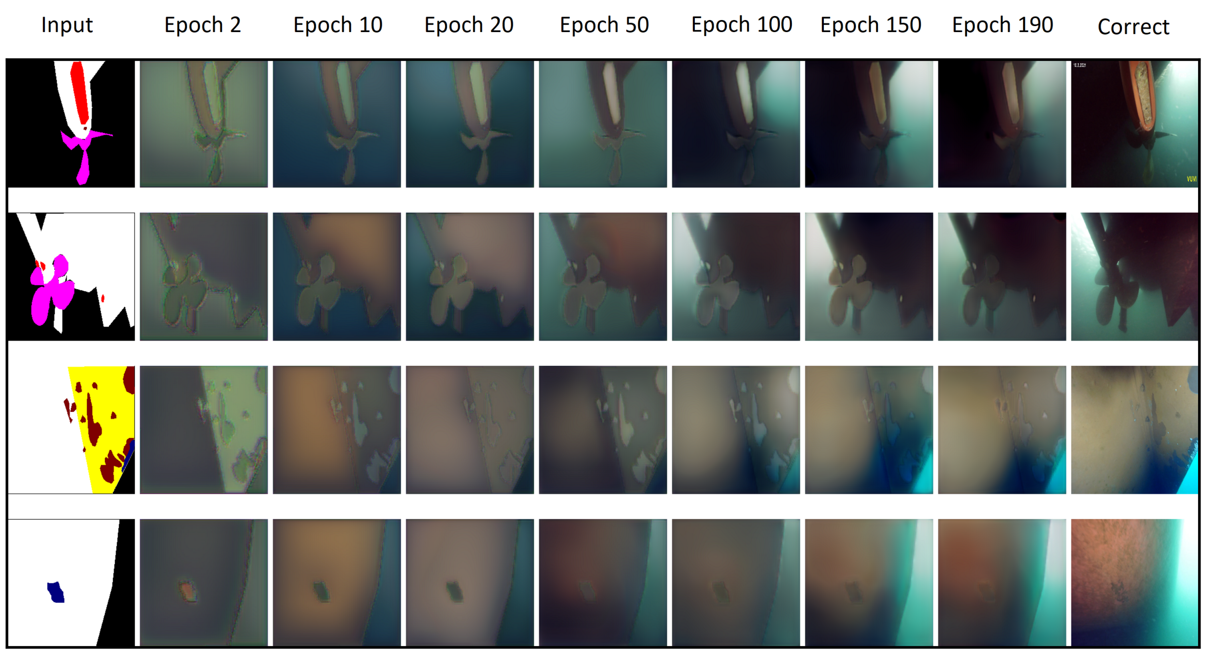
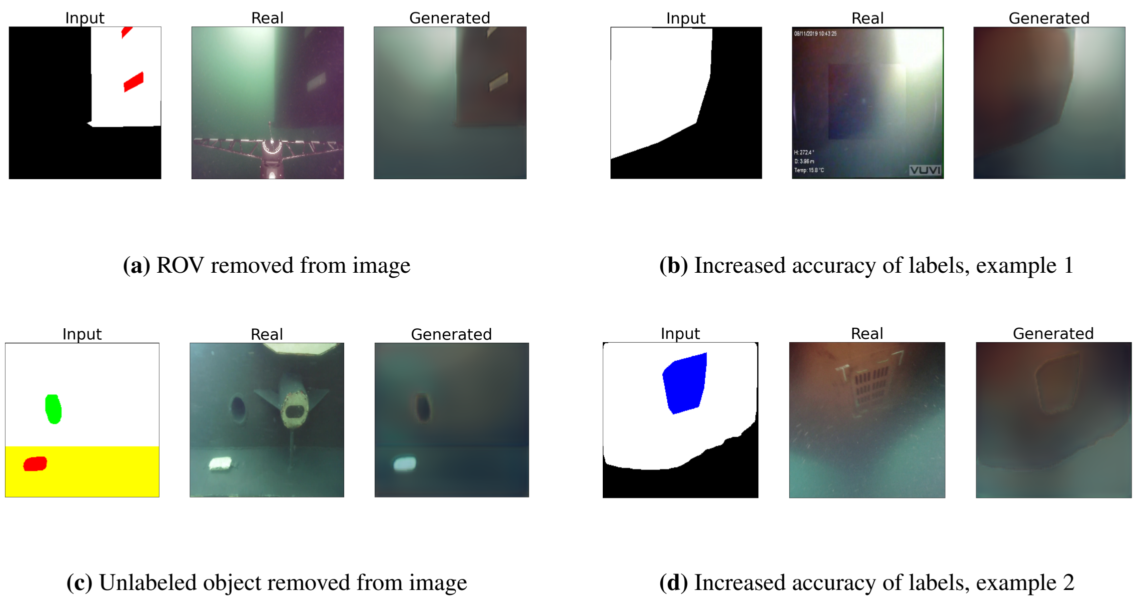


Figure 4.6: Development of predicted images during training of cGAN

Figure 4.6 shows the process of training ten Pix2Pix-cGAN over 190 epochs. The generated images becomes more and more realistic, where the cGAN adds atmospheric light and reflected light from the ROV to the image. This shows that the cGAN have the ability to generate semi-realistic images with different classes, determined by the input.



(a) ROV removed from image

(b) Increased accuracy of labels, example 1

(c) Unlabeled object removed from image

(d) Increased accuracy of labels, example 2

Figure 4.7: Possible advantages with generated images

Table 4.5: SUIM-Net trained on a standard training set and SUIM-Net trained on a dataset with real and generated images

Dataset	Anode	Bilge keel	Corrosion	Defect	Marine growth	Valves	Paint peel	Propeller	Sea chest grating	Ship hull	Average
Real	40.5	47.7	17.2	7.9	44.1	30.8	25.9	71.3	79.9	80.0	44.5
Real + generated	33.8	41.9	13.1	0.72	44.6	30.1	22.0	65.4	81.2	80.2	41.3

Table 4.5 shows a comparison between the SUIM-Net trained with a normal training set and a training set where a generated version of all images are added to the training set. This test is done to see if advantages from generated images can be utilized in the learning process. Examples of possible advantages are shown in Figure 4.7. Figure 4.7b and Figure 4.7d shows examples where the generated images will match the labels better. The generated images are based on the labels and therefore will match the labels with a high accuracy. The ship hull will end exactly where the ship hull ends in the mask, which is not always the case with the real image. Figure 4.7a and Figure 4.7c shows examples where the generated image removes objects that are not present in the segmentation mask. This can help in the training by letting the model focus on the objects it is supposed to learn without distractions.

From the results in Table 4.5, it is clear that the generated images will cause a worse performance. This can be because the generated images do not add any additional information, or that the GAN is not creating realistic enough images with enough detail.

Table 4.6: Five-fold cross-validation of SUIM-Net$_{bn+do}$ trained on a small training set and SUIM-Net$_{bn+do}$ trained on a small training set augmented with generated data

Dataset	Anode	Bilge keel	Corrosion	Defect	Marine growth	Valves	Paint peel	Propeller	Sea chest grating	Ship hull	Average
0.5 Real	34.4	17.4	1.0	0.4	19.8	32.6	17.0	43.4	70.5	77.2	31.4
0.5 Real + 0.5 generated	34.8	27.8	4.9	1.5	38.3	42.1	19.3	59.9	70.9	79.3	37.9

In Table 4.6 a five-fold cross-validation is done to check another aspect of augmenting the training set. In this comparison, the cGAN is using segmentation masks from images not present in the training set. This is done by splitting the total training set in half. The first half is added to a training set, called "0.5 Real", and used for the baseline. This baseline is tested against SUIM-Net trained on the same half, but with added generated images based on the segmentation masks in the second half of the total training set. This new augmented training set is called "0.5 Real + 0.5 generated" in the table. Table 4.6 shows significant better results when the dataset is augmented with generated images. The performance is increased with 20.7%. This proves the possibility of increasing the performance of the segmentation models by augmenting the dataset with new masks. This can help to increase the prediction performance of classes with poor representation, such as the "defect" class. One issue is that if the defect class have a poor representation in the dataset, it can be difficult for the cGAN to learn how to generate synthetic versions representations of the class.

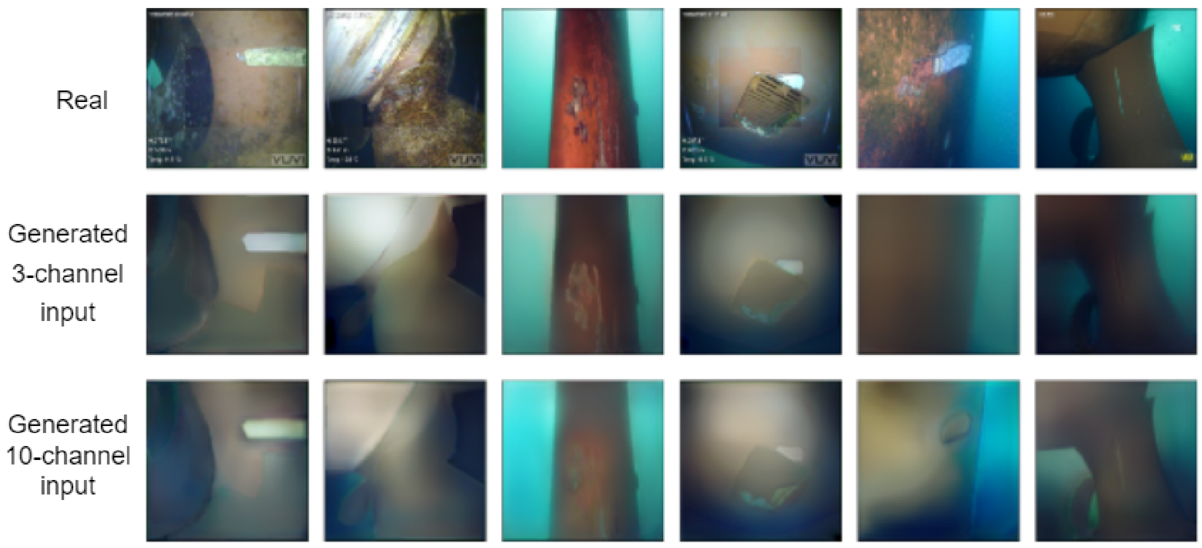


Figure 4.8: Real image and corresponding generated images

Figure 4.8 Shows generated images from the cGAN based on a 3-channel input and the CGAN based on a 10-channel input. The cGAN with a 10-channel input generates equally realistic images as the 3-channel version and can probably be used in the same way as the 3-channel GAN. The cGAN with 10-channel input will receive more information about the scene, since when two labels are overlapping, one of them will be removed in the 3-channel label. The 10-channel cGAN has therefore a larger potential in generating high-quality images.

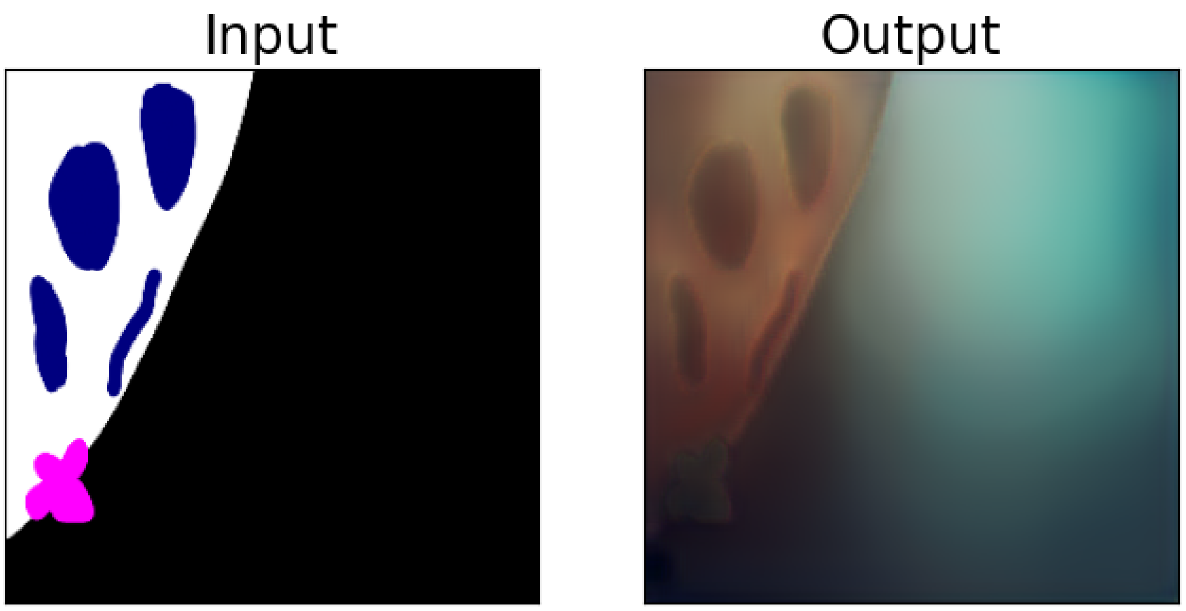


Figure 4.9: Synthetic generated image with synthetic 3-channel label input

Figure 4.9 shows one use-case of the cGAN. Here, a 3-channel label is used as input, where the label is not based on any real data. The cGAN outputs an image with realistic backlight and a reddish ship hull. The propeller is barely visible since the propeller is located at the bottom, where little light normally is present. Marine growth is also present in the image, but is not as realistic compared to real marine growth. The generated image does not contain enough texture and details for a realistic appearance.

4.4 Video segmentation

In the video analysis, the chosen SUM-Net$_{a}tt$ model will be tested together with the weighted average image stitching algorithm for better and more constant predictions. There are no ground truth segmentation labels to compare with here, so the focus will be on how the weighted average will affect the predictions. These results were retrieved at the same time as the enhancement methods were tested, so the results in this chapter will be without the proposed enhancement method. In this section, example frames are shown of the video segmentation.

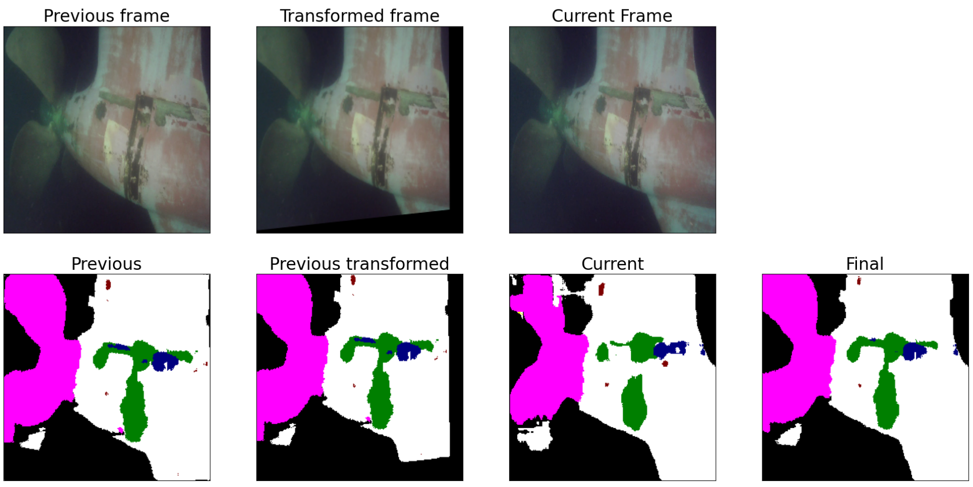


Figure 4.10: Weighted average with good match

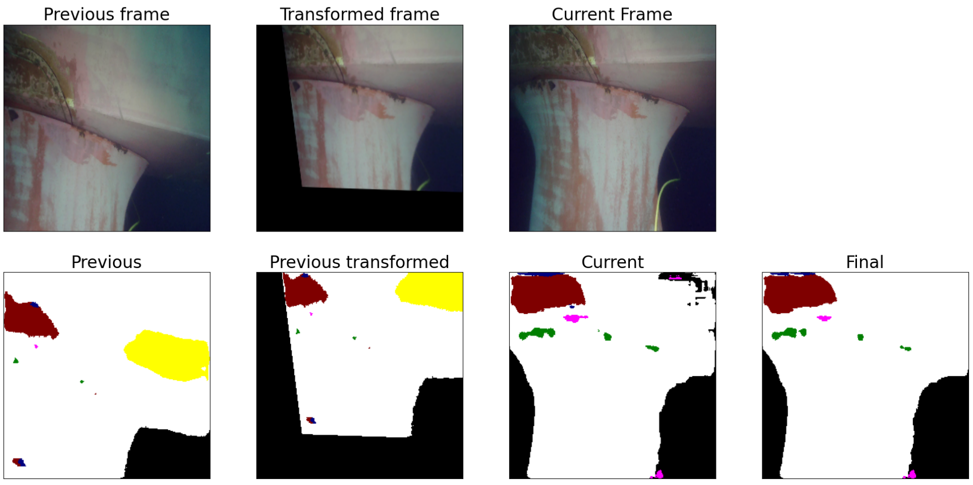


Figure 4.11: Weighted average with poor match

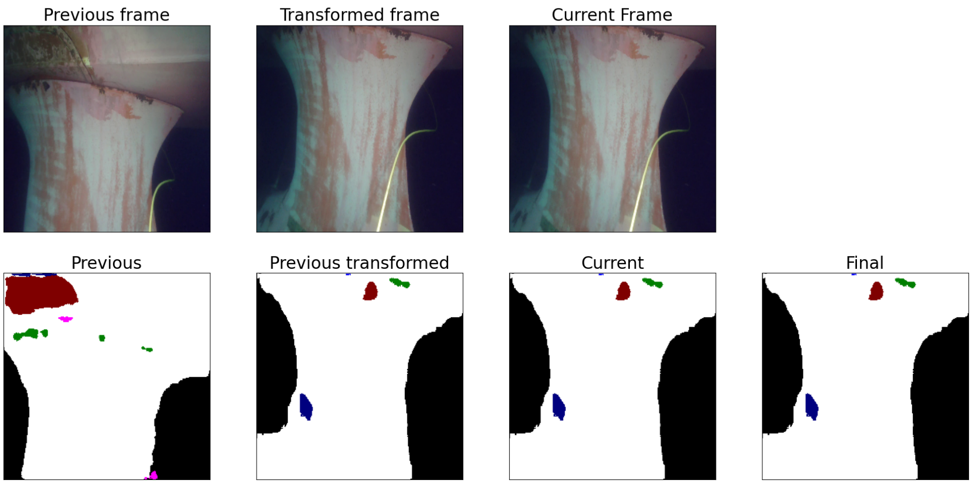


Figure 4.12: Weighted average with no match

Figure 4.10, Figure 4.10 and Figure 4.10 shows three different cases that can occur when analyzing frame by frame with the image averaging algorithm. To make the results more clear, 30 frames are skipped between each analyzed frame. Figure 4.10 shows an example classified as an accurate transformation. In this example, 102 good matches have been detected, and the weighing becomes 0.25 which means the "previous transformed" is weighted four times as much as the new classification. One can see that the current prediction is adjusted by the previous transformed in the final prediction, where errors from the current prediction have been removed, while new information in the right and bottom have been added. Figure 4.11 is classified as a poor transformation match, where only 24 matches found. Here, the weight is 1.78, which means the new prediction is weighted 1.78 times higher than the weighted average prediction from the previous frame. In Figure 4.12, the number of good matches is three, which is below the threshold for using the average. This will result in that the previous prediction is disregarded, and the current prediction replaces the previous transformed in the further calculations, which is shown in the figure.

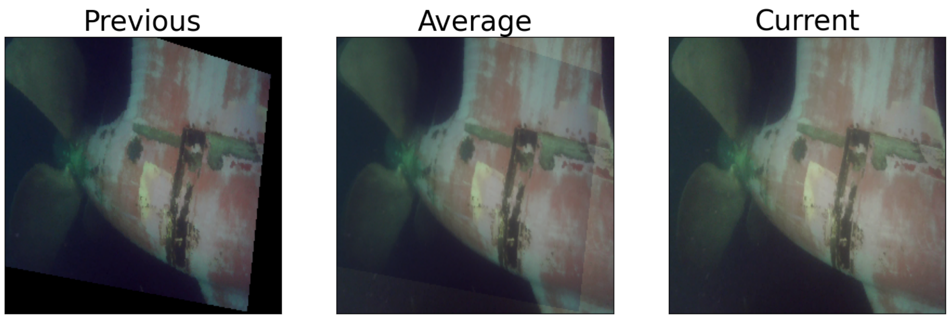


Figure 4.13: Two frames and the average

Figure 4.13 shows two frames and the average of the two images where overlapping occurs with a weighing of one. Only continuous prediction masks are transformed and averaged in the video analysis algorithm, but this can give a better visualization of the idea behind the weighted average. An image can appear blurry when averaged like this, but the same effect is utilized to create predictions with fewer variations by for example removing some wrong estimates caused by particles in the water.

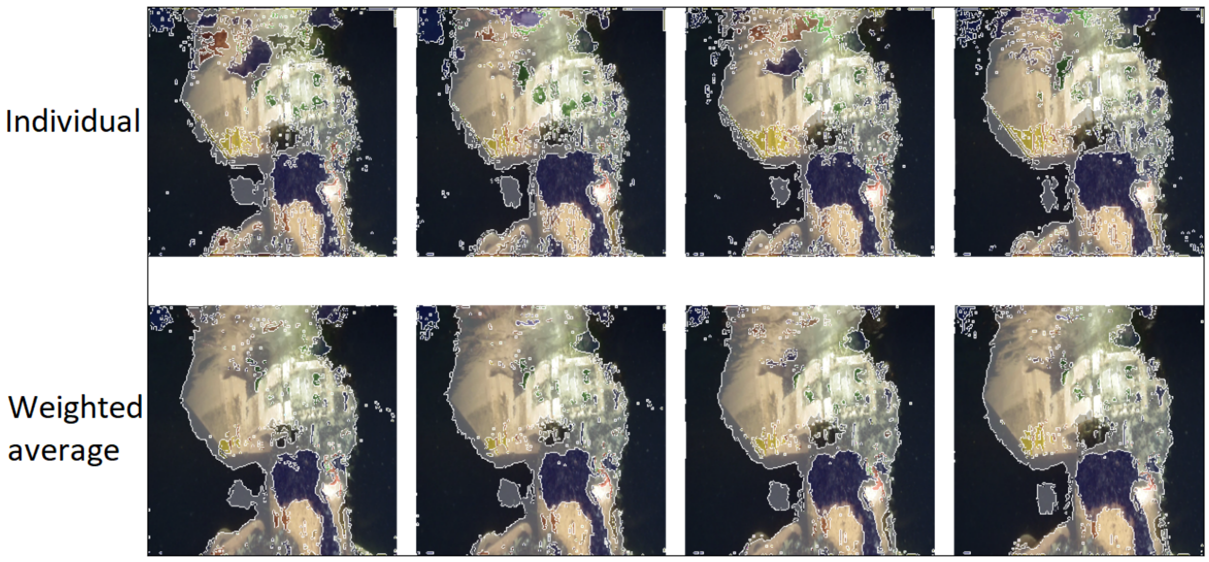


Figure 4.14: Independent segmentation and weighted average segmentation

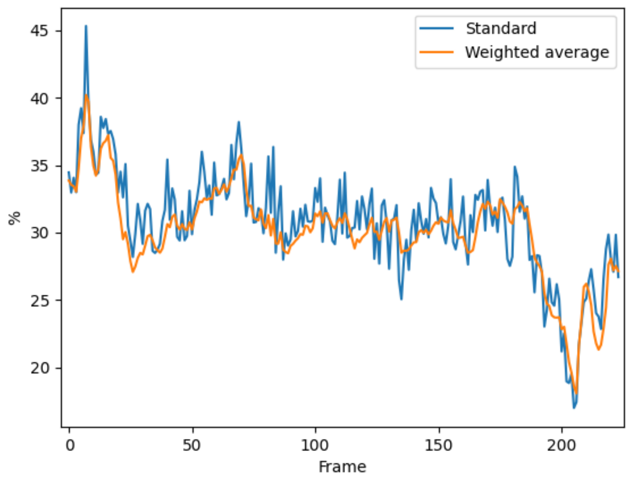


Figure 4.15: Positive predictions of the class "marine growth" frame by frame

Figure 4.14 Shows a comparison of four consecutive frames, where the segmentation is added to the image. In this comparison, one can see the effect of the weighted average where there are less noise and variation in the predictions done with the weighted average. This is shown in Figure 4.15, where the predictions of "marine growth" will vary less with a weighted average. The predictions will gradually change instead of jumping back and forth independently of the previous prediction. A gradual change is desired since the field of view of the camera will change gradually during an inspection, which means the presence of different classes will change slowly frame by frame. In addition, one can see in Figure 4.14 that there are less small white dots in the weighted average, especially in the frame to the right. These white dots are errors, possibly caused by particles in the water, and the weighted average will act as a filter and remove some of these error predictions.

Table 4.7: Variance of positive predictions during a video

Method	Anode	Bilge keel	Corrosion	Defect	Marine growth	Valves	Paint peel	Sea chest grating	Ship hull	Average
Standard	0.0096	0.0007	0.0869	0	15.7	0	55.37	0.32	0.00081	24.86
Weighted average	0.0004	0.0000	0.0004	0	12.9	0	0.37	0.10	0.0006	24.32

Table 4.7 shows the variance of all classes in the same video used to plot Figure 4.15. All variances are lower with the weighted average. In the video, no bilge keel are present. Since the variance is not zero for the standard method, some false positive predictions are present. With the weighted average, the variance is zero, which means that the false positive prediction have been filtered out. This indicates another performance improving aspect of the weighted average.

4.4.1 Final Video Analysis

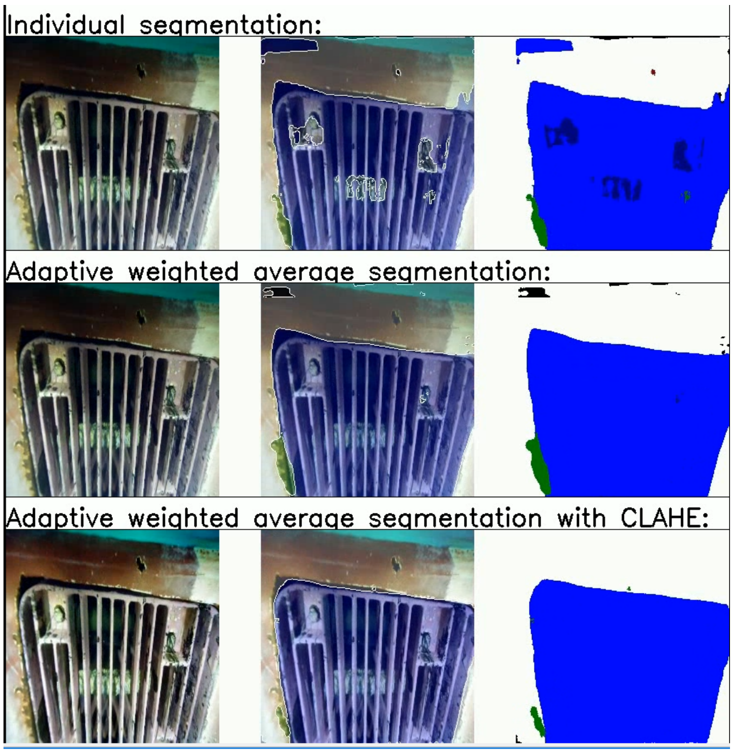

Individual segmentation:
Adaptive weighted average segmentation:
Adaptive weighted average segmentation with CLAHE:

Figure 4.16: Frame from final video segmentation

Figure 4.16 shows an example frame from the video segmentation, where ship hull, sea chest grating and corrosion is detected. Three methods are used in the video to highlight advantages and differences of the methods. The upper methods show video segmentation where all frames are evaluated independently. The middle frame shows video segmentation combined with the adaptive weighted average computer vision method. The bottom method combines the middle method CLAHE enhancement. CLAHE was chosen based on results from section 4.2. In this frame example, CLAHE predicts the location of the ship hull the best, while individual segmentation have false positive predictions of marine growth. Four videos from ship inspections are used to show how video segmentation will perform in a real life scenario. The method for visualizing the prediction results and color codes used is described in section 3.7.1. The four videos are chosen to show video segmentation of different aspects of the underwater ship inspection. It is recommended to look at the videos for a full understanding of the performance of the video segmentation.

Video 1, available at: `https://youtu.be/QsDGcyKQP1Q`,shows an inspection of a propeller pod, where classes such as hull, propeller, corrosion, paint peel and possibly marine growth are present. Towards the end of the video (45 seconds), CLAHE does not have false predictions of the propeller, while the other methods do.

Video 2, available at: `https://youtu.be/xttT9I67Xr8`, shows an inspection of an azimuth propeller with relatively much marine growth, corrosion, and paint peel. The averaging in the bottom two methods performs well in the aspect of from the individual segmentation.

Video 3, available at: `https://youtu.be/6b4hjV8SX3g`, shows inspection of sea chest grating. The models predict well from a distance. Close to the gratings, the predictions become poorer. Corrosion is also predicted to be present at one of the gratings inspected, which looks probable.

Video 4, available at: `https://youtu.be/fvMjTieAvPI`, shows an inspection along a ship hull, where the bilge keel is present at the bottom of the hull. The method with CLAHE locates more of the paint peel locations, but have also some areas where the ship hull is not located correctly. CLAHE seems also more consistent in the predictions of the paint peel (1:15 to 1:20), while the upper and middle methods alternates between predicting marine growth.

From the videos, it can be seen that the computer vision method will act as a filter and remove noise from the predictions. The location and shape of the predictions is also more consistent. Combining the computer vision method with the enhancement method gave a higher detectability rate of classes with less visibility in the images.

Chapter 5

Discussion

This section provides a further discussion of the results and analysis in chapter 4 and discusses inconsistencies and other aspects that will have an influence on the performance and evaluation.

5.1 Data for Evaluation and Training

5.1.1 Dataset

The LIACI dataset was used for all training and quantitative evaluation of the segmentation models implemented in this thesis. The LIACI dataset can be classified as a large scale dataset, with 1893 images, but will still only represent a small area of the domain that can be encountered. The images in the LIACI dataset are based on images from 16 different ship inspections, which will create a variety of the annotated classes, but will not create a full representation of all variations that can be encountered during a ship inspection in regard to the age of the ship, the color of the ship and the underwater scene to mention a few. Variations in cameras used for the inspection can also cause differences, where in the LIACI dataset, four different cameras are used. The goal through deep learning is to create generalized mappings, but this cannot be verified to a full extent when data from this dataset is used for both training and quantitative evaluation. The evaluations are done by randomly splitting the 1893 images from the dataset into a training set with 1515 images in the training set, 189 images in the validation set and 189 images in a test set with the corresponding masks. This is done to avoid as much as possible of the overfitting effects to affect the evaluation. When creating sets randomly, the test set can be a poor representation of the training set. Therefore, in the more comprehensive comparisons in this thesis, each model is trained five times on randomly composed sets. A seed is used to generate the same random sets for each model. Over performing can still occur as an effect of overfitting, since the dataset consists of images from video sequences. The test set and validation set will contain images similar to the images in the training set. As observed in Figure 4.1 overfitting occurs during training. The validation set is therefore used as a generalization metric to only save when the performance increases on the validation set, but overfitting can still affect the evaluation.

The LIACI set is annotated manually, which can be difficult with poor visibility. The classes in the dataset can also be difficult to separate for an annotator, and errors in the annotations can occur, shown in Figure 4.2, where a ship hull seems to be present, but not annotated in the label masks. The model trained can only be as good as the accuracy and consistency of the dataset trained on, so this will be a limiting factor of the performance.

5.1.2 Videos

The videos used in this project can give a less biased evaluation than the results from the test sets, but the prediction performance of videos cannot be evaluated by quantitative measures directly, since no ground truth exists. Therefore, a visual evaluation is needed for this type of evaluations. One quantitative measure of the videos, that is used in the results, is the variance of the class predictions. In a video filmed at slow speed, the classes present in the field of view will change gradually, so a low variance in the predictions will give indications of a better performance, but must be validated by checking if a lower variance is caused by filtering effects, which is desired. An example of this is shown in Figure 4.15. The videos are also used for quantitative evaluation, by looking at the number of good features matched between images, which is used as a metric for the quality of a transformation. More matched features will probably mean a good match, and can increase the quality of the homography matrix.

5.2 Model Comparison

The standard deviation in Table 4.1 gives no information about if the higher performance is statistically significant, since the variance describes the difference of different tests, where the performance will differ. If standard deviation were to be used for this purpose, training had to be performed several times on one test and not one time on several tests. This could have given a better indicator of the performance differences, but is not done due to time constraints. In the model comparisons, only models with VGG16 as a backbone are tested, and this is done for more comparable results and to narrow the model testing part of the project. Because of the time constraints, this thesis is limited to few varieties of models. Other models, for example SOTA models designed for real-time segmentation, can be more suited.

Several of the model comparisons are trained on one training set. This gives more room for inaccurate and less conclusive results than comparing with a five-fold cross-validation. Comparing models trained on one training set will still give indicative results of the performance of the model compared.

In Table 4.2, the PSP-Net gets a much lower score than what could be anticipated. This could indicate some errors in the implementation of the model. The performance of the PSP-Net were good during training, both on the training set and the validation set. Proof of this is not shown in the results. The good performance during training could indicate that an error had occurred during IoU-evaluation, but Figure 4.4 shows poor visual results too.

A decrease in validation loss was used to determine when to save the weights during training of the models. Since the IoU score is used for comparison, it was tested to use the IoU score to use the IoU score to determine when to save the model during training and as a loss function. These tests did not result in a better IoU score on the test set, so binary was used for both these purposes in this thesis.

5.3 Image Enhancement

Image enhancement is originally investigated to improve semantic segmentation, and the improvement of feature matching is an additional use case, where other image pre-processing methods can give better image matching performance.

The GAN-based image enhancement method had generally worse performance than the conventional enhancement methods. One reason for this is that they are trained to improve an image based on learned filters, which is what the decoder of the SUIM-Net is doing as well. Instead of the learning what filters to be applied, it has to learn what filters that has been applied in the pre-processing. One other issues with the GAN-based models is that they are trained on other datasets, where these datasets can be too different from the LIACI dataset, which can cause poor attempts of image enhancing and confusing the segmentation model. For increased feature matching, the performance of the GAN models were poor. The GAN models will alter the image based on a high dimensional, unpredictable mapping, which can cause the ORB to create different descriptors at the same location in different frames. ORB needs consistency to work properly.

5.4 Generated Images

The cGAN implemented in this project showed some promising results by augmenting a training set based on labels not present in the set. The cGANs were implemented to look at possible use cases, but the training setup were not prioritized because of time limitations. This part of the project acted more as a supplement for testing alternative ways to improve the training on the data set, but interesting results were still found. During training, the discriminator reached a too high accuracy, which can be looked at as a form of overfitting and slowed the learning of the generator. Other capabilities of the cGAN, like removing unlabeled information and adjusting the image to fit the label, did not provide any improvements, while it is considered to be possible. The implemented cGAN with 10-channel input was not either utilized, but is considered as better suited to the LIACI dataset, since it uses the same dimensions as the segmentation models does.

5.5 Video Analysis

One error with the adaptive weighted average is that when transforming the images, a white border can appear around the transformed images. This is not seen in the results, but will appear in the videos when averaging the predictions. Some of this issue is removed by ignoring prediction values close to one in the transformed predictions by using a threshold. Setting this value too low will also remove very certain predictions, which is not desired.

A challenging aspect of working with videos is that the results are not directly displayable in a report. In the results, figures displaying frames and consecutive frames are used to highlight the main aspects evaluated during the video analysis. Plots and tables have been used to concretize some aspects of the live segmentation, but visual, subjective evaluation of the results is needed for a full evaluation.

The final run-time of the method is not fast enough for a high frames-per-second estimation. Because of this, the absolute run-time is not considered during the result analysis, while relative run-times are weighted, since the use case is live-segmentation. Optimizing code for efficiency demands much work and would limit other aspects of this thesis.

The segmentation videos lost some quality when being uploaded to YouTube, and higher quality videos can be found in the delivered materials. When the videos were segmented, only every fifth frame were used for segmentation. Using every frame could have resulted in even better performance of the weighted average algorithm. Using the segmentation videos for a visual evaluation is challenging because, as mentioned in Waszak et al. (In Press), it is difficult to separate the classes “corrosion”, “paint peel” and “marine growth”. One takeaway is that the automatic segmentation will give relatively accurate results that one of the classes is present, which is a good indication of degradation on the ship hull.

Chapter 6

Conclusion and Further Work

6.1 Conclusion

In this thesis, a method is proposed to improve segmentation performance for underwater ship inspection by using the LIACI dataset for training. Three different approaches have been done and tested for increased performance.

The first objective was to find a suitable model for real-time semantic segmentation in an underwater environment, where both the prediction score and the network size are the two factors considered. The SUM-Net was chosen as the network to use because of its performance on a similar real-time underwater case. Different versions of the SUIM-Net were implemented and compared, where the proposed SUIM-Net$_{att}$ showed better results with low increased run-time. The SUIM-Net$_{att}$ is therefore the preferred model for this project. The SUIM-Net$_{att}$ was tested against three other SOTA models: U-Net, PSP-Net, and DeepLabv3+, where the SUIM-Net$_{att}$ achieved the best results.

The second objective was to use image enhancement methods, focusing on underwater enhancement methods, to increase the segmentation and feature matching performance. Seven existing methods were tested and compared for this purpose. The test showed that the UCM and the CLAHE methods increased the performance significantly in both comparisons. Both methods are viable options for increasing the performance of segmentation and feature matching, but CLAHE was chosen for the final model because of the negligible run-time.

A Pix2Pix was implemented and trained to investigate if augmenting the dataset with generated images would increase the performance of the segmentation when trained on the augmented dataset. The two versions of Pix2Pix generated images that had a realistic appearance, but without too much detail. The performance did not increase when the dataset was augmented based on masks already in the dataset. When augmenting the training set based on masks not present in the training set, the performance increased significantly and is considered a promising solution for increasing the performance.

For video segmentation, a weighted average is implemented to combine the previous transformed prediction with the current prediction to utilize the continuous previous predictions to create a less varying and more precise prediction. This method was shown to give significantly better results than only using the new prediction independently of the previous prediction.

For a final framework for live semantic segmentation during an underwater ship inspection, it is proposed, based on the results, to use SUIM-Net$_{att}$ as the model, enhance the images with CLAHE and use the adaptive weighted average to combine the segmentation predictions. The final combined model was applied to four underwater video inspections, where the final model showed a better performance. The largest improvement was caused by adding the adaptive weighted average method.

The use of videos for testing showed the applicability of the proposed method for automated underwater ship inspections.

6.2 Further Work

This thesis covers several research fields such as deep learning, computer vision, and image enhancement. The results showed promising results for automatic estimation for locating objects of interest during a ship inspection, and several aspects investigated in this thesis can be improved and further developed. Further work is recommended in the following:

- Look at other deep learning models beyond the models in the presented material, with more comprehensive comparisons.
- Look further into the performance-increasing effect of using image enhancement methods as a pre-processing step for image matching and semantic segmentation.
- Improve the adaptive weighted average image matching algorithm for more precise object location during a ship inspection. A more accurate weight function and other forms of determining the quality of the transformation matrix between frames is recommended.
- Investigate if GANs be used to augment the dataset to improve the learning capabilities of poorly represented classes, based on constructed labels, or by generating more accurate images based on the existing image-mask pairs.
- Improve the prediction averaging algorithm by using sensor inputs other than camera inputs, such as sensor information about velocity, acceleration, and relative position to the field-of-vied.
- Extend the method for a complete automatic inspection by mapping the segmentation estimation from the camera to a position, by the use of visual 3D-reconstruction and mapping techniques such as SLAM, dimension information about the specific ship used in the inspection or position information from other sensors.
- Look further into the improvement of frame-by-frame segmentation with for example the use of RCNNs. This would require a dataset structured with complete image sequences and corresponding label masks.

Bibliography

Albawi, S., Mohammed, T.A., Al-Zawi, S., 2017. Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology (ICET), pp. 1–6. doi:10.1109/ICEngTechnol.2017.8308186.

Alsallakh, B., Kokhlikyan, N., Miglani, V., Yuan, J., Reblitz-Richardson, O., 2020. Mind the pad–cnns can develop blind spots. arXiv preprint arXiv:2010.02178 doi:10.48550/arXiv.2010.02178.

Ancuti, C., Ancuti, C.O., Haber, T., Bekaert, P., 2012. Enhancing underwater images and videos by fusion, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 81–88. doi:10.1109/CVPR.2012.6247661.

Antony Jacob, A., 2018. Underwater inspection in lieu of dry-docking (uwild) using remotely operated vehicles (rov).

Attention UNet, 2019, . URL: https://github.com/MoleImg/Attention_UNet. accessed: 2022-26-04.

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., 2008. Speeded-up robust features (surf). Computer Vision and Image Understanding 110, 346–359. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555, doi:https://doi.org/10.1016/j.cviu.2007.09.014. similarity Matching in Computer Vision and Multimedia.

Baykal, S.I., Bulut, D., Sahingoz, O.K., 2018. Comparing deep learning performance on bigdata by using cpus and gpus, in: 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), pp. 1–6. doi:10.1109/EBBT.2018.8391429.

Benardos, P., Vosniakos, G.C., 2007. Optimizing feedforward artificial neural network architecture. Engineering Applications of Artificial Intelligence 20, 365–382. URL: https://www.sciencedirect.com/science/article/pii/S0952197606001072.

BUBER, E., DIRI, B., 2018. Performance analysis and cpu vs gpu comparison for deep learning, in: 2018 6th International Conference on Control Engineering Information Technology (CEIT), pp. 1–6. doi:10.1109/CEIT.2018.8751930.

Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A., 2020. Albumentations: Fast and flexible image augmentations. Information 11. URL: https://www.mdpi.com/2078-2489/11/2/125, doi:10.3390/info11020125.

Calonder, M., Lepetit, V., Strecha, C., Fua, P., 2010. Brief: Binary robust independent elementary features, in: Daniilidis, K., Maragos, P., Paragios, N. (Eds.), Computer Vision – ECCV 2010, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 778–792.

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

Chigozie Nwankpa, Winifred Ijomah, A.G., Marshall, S., 2018. Activation functions: Comparison of trends in practice and research for deep learning. IEEE Transactions on Neural Networks URL: https://arxiv.org/abs/1811.03378v1.

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

Costa, H., Iñarra Abad, S., 2019. Symbiosis between panoramic photography, cgi and freehand drawing in landscape representation 12, 4.1 – 4.11.

DNV, 2022a. Corrosion.ai, corrosion inspection solution. URL: https://www.dnv.com/research/review-2020/featured-projects/corrosion-ai-inspection.html#. accessed: 2022-06-06.

DNV, 2022b. Overview of the carbon intensity indicator. URL: https://www.dnv.com/maritime/insights/topics/CII-carbon-intensity-indicator/index.html. accessed: 2022-06-06.

Elvesæter, B., 2022. Liaci - lifecycle inspection, analysis and condition information system. URL: https://www.sintef.no/en/projects/2021/liaci/. accessed: 2022-06-02.

Fanebust, F., Smith-Solbakken, M., 2021. Nordsjødykkerne. URL: https://snl.no/nordsj%C3%B8dykkerne. accessed: 2021-12-12.

Gatys, L.A., Ecker, A.S., Bethge, M., 2016. Image style transfer using convolutional neural networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2414–2423. doi:10.1109/CVPR.2016.265.

Gholamalinejad, H., Khosravi, H., 2020. Pooling methods in deep neural networks, a review.

Glorot, X., Bordes, A., Bengio, Y., 2011. Deep sparse rectifier neural networks, in: AISTATS.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial networks. Advances in Neural Information Processing Systems 3. doi:10.1145/3422622.

Google Colab, 2022, . URL: https://colab.research.google.com/. accessed: 2022-06-02.

Guo, Y., Li, H., Zhuang, P., 2020. Underwater image enhancement using a multiscale dense generative adversarial network. IEEE Journal of Oceanic Engineering 45, 862–870. doi:10.1109/JOE.2019.2911447.

Hakim, M., Utama, I.K., Nugroho, B., Yusim, A., Baithal, M., Suastika, K., 2018. Review of correlation between marine fouling and fuel consumption on a ship.

HDF5, 2022, . URL: https://www.hdfgroup.org/. accessed: 2022-11-05.

He, K., Sun, J., Tang, X., 2009. Single image haze removal using dark channel prior, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1956–1963. doi:10.1109/CVPR.2009.5206515.

He, K., Zhang, X., Ren, S., Sun, J., 2016a. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

He, K., Zhang, X., Ren, S., Sun, J., 2016b. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

Hover, F.S., Eustice, R.M., Kim, A., Englot, B., Johannsson, H., Kaess, M., Leonard, J.J., 2012. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. The International Journal of Robotics Research 31, 1445–1464. doi:10.1177/0278364912461059.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 .

Hu, K., Weng, C., Zhang, Y., Jin, J., Xia, Q., 2022. An overview of underwater vision enhancement: From traditional methods to recent deep learning. Journal of Marine Science and Engineering 10. URL: https://www.mdpi.com/2077-1312/10/2/241, doi:10.3390/jmse10020241.

Huang, G.B., 2003. Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Transactions on Neural Networks 14, 274–281. URL: https://10.1109/TNN.2003.809401.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift .

Iqbal, K., Abdul Salam, R., Azam, O., Talib, A., 2007. Underwater image enhancement using an integrated colour model. IAENG International Journal of Computer Science 34.

Iqbal, K., Odetayo, M., James, A., Salam, R.A., Talib, A.Z.H., 2010. Enhancing the low quality images using unsupervised colour correction method, in: 2010 IEEE International Conference on Systems, Man and Cybernetics, pp. 1703–1709. doi:10.1109/ICSMC.2010.5642311.

Islam, M.J., Edge, C., Xiao, Y., Luo, P., Mehtaz, M., Morse, C., Enan, S.S., Sattar, J., 2020a. Semantic segmentation of underwater imagery: Dataset and benchmark, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 1769–1776.

Islam, M.J., Luo, P., Sattar, J., 2020b. Simultaneous enhancement and super-resolution of underwater imagery for improved visual perception .

Islam, M.J., Xia, Y., Sattar, J., 2020c. Fast underwater image enhancement for improved visual perception. IEEE Robotics and Automation Letters 5, 3227–3234. doi:10.1109/LRA.2020.2974710.

Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A., 2017. Image-to-image translation with conditional adversarial networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5967–5976. doi:10.1109/CVPR.2017.632.

Kamann, C., Rother, C., 2020. Benchmarking the robustness of semantic segmentation models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Keras, 2022, 2022. URL: https://keras.io/. accessed: 2022-01-20.

Keras HDF5 ImageDataGenerator, 2020, . URL: https://github.com/angulartist/Keras-HDF5-ImageDataGenerator. accessed: 2022-01-03.

Kim, J., Kim, J., Choi, J., 2021. Sequential movie genre prediction using average transition probability with clustering. Applied Sciences 11. URL: `https://www.mdpi.com/2076-3417/11/24/11841`, doi:`10.3390/app112411841`.

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

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. IEEE Transactions on Neural Networks 521, 436—-444. doi:`10.1038/nature14539`.

Leutenegger, S., Chli, M., Siegwart, R.Y., 2011. Brisk: Binary robust invariant scalable keypoints, in: 2011 International Conference on Computer Vision, pp. 2548–2555. doi:`10.1109/ICCV.2011.6126542`.

Li, D., Yang, J., Kreis, K., Torralba, A., Fidler, S., 2021. Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8296–8307. doi:`10.1109/CVPR46437.2021.00820`.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988.

Linder-Norén, E., 2019. Keras-gan. URL: `https://github.com/eriklindernoren/Keras-GAN`. accessed: 2022-23-04.

Liu, S., 2018. Udid dataset. URL: `http://vision.is.tohoku.ac.jp/~liushuang/a-vision-based-underwater-docking-system/dataset/`. accessed: 2021-19-11.

Liu, S., Ozay, M., Okatani, T., Xu, H., Lin, Y., Gu, H., 2018. Learning deep representations and detection of docking stations using underwater imaging, in: 2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO), pp. 1–5. doi:`10.1109/OCEANSKOBE.2018.8559067`.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431–3440. doi:`10.1109/CVPR.2015.7298965`.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60. URL: `10.1023/B:VISI.0000029664.99615.94`.

Luong, M.T., Pham, H., Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 .

Madani, A., Moradi, M., Karargyris, A., Syeda-Mahmood, T., 2018. Semi-supervised learning with generative adversarial networks for chest x-ray classification with ability of data domain adaptation, in: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), pp. 1038–1042. doi:`10.1109/ISBI.2018.8363749`.

Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets .

Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y., 2018. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 .

Mo, Y., Wu, Y., Yang, X., Liu, F., Liao, Y., 2022. Review the state-of-the-art technologies of semantic segmentation based on deep learning. Neurocomputing 493, 626–646. URL: `https://www.sciencedirect.com/science/article/pii/S0925231222000054`, doi:`https://doi.org/10.1016/j.neucom.2022.01.005`.

Nair, V., Hinton, G., 2010. Rectified linear units improve restricted boltzmann machines vinod nair, pp. 807–814.

O'Byrne, M., Pakrashi, V., Schoefs, F., Ghosh, B., 2018. Semantic segmentation of underwater imagery using deep networks trained on synthetic imagery. Journal of Marine Science and Engineering 6, 93. doi:10.3390/jmse6030093.

Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., et al., 2018. Attention u-net: Learning where to look for the pancreas. arXiv preprint arXiv:1804.03999 .

OpenCV, 2022, . URL: https://opencv.org/. accessed: 2022-08-03.

PaperSpace, 2022, . URL: https://www.paperspace.com/. accessed: 2022-21-02.

Peng, D., Zhang, Y., Guan, 2019. End-to-end change detection for high resolution satellite images using improved unet++. Remote Sensing 11, 1382. doi:10.3390/rs11111382.

Petricca, L., Moss, T., Figueroa, G., Broen, S., 2016. Corrosion detection using a.i : A comparison of standard computer vision techniques and deep learning model, pp. 91–99. doi:10.5121/csit.2016.60608.

Pranoto, H., Budiharto, W., Hendric Spits Warnars, H.L., Matsuo, T., Heryadi, Y., 2018. Image size, color depth, age variant on convolution neural network, in: 2018 Indonesian Association for Pattern Recognition International Conference (INAPR), pp. 39–45. doi:10.1109/INAPR.2018.8627054.

PyTorch, 2022, . URL: https://pytorch.org/. accessed: 2022-01-20.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, pp. 234–241. doi:10.1007/978-3-319-24574-4_28.

Rublee, E., Rabaud, V., Konolige, K., Bradski, G., 2011. Orb: An efficient alternative to sift or surf, in: 2011 International Conference on Computer Vision, pp. 2564–2571. doi:10.1109/ICCV.2011.6126544.

Russell, S., Norvig, P., 2019. Artificial Intelligence: A Modern Approach, third edition. PEARSON. URL: http://aima.cs.berkeley.edu.

Scherer, D., Müller, A., Behnke, S., 2010. Evaluation of pooling operations in convolutional architectures for object recognition, pp. 92–101. doi:10.1007/978-3-642-15825-4_10.

Schlemper, J., Oktay, O., Schaap, M., Heinrich, M., Kainz, B., Glocker, B., Rueckert, D., 2019. Attention gated networks: Learning to leverage salient regions in medical images. Medical image analysis 53, 197–207.

Scope Group, 2021. Scope launches ship review: 70,000 esg ship-assessments, 40,000 cii ratings. URL: shorturl.at/qDLNX. accessed: 2022-06-06.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv 1409.1556 URL: https://arxiv.org/abs/1409.1556v6.

Smith, S.W., 1997. The scientist and engineer's guide to digital signal processing. California Technical Publishing, San Diego.

Spilsbury, T., Camps, P., 2019. Don't ignore dropout in fully convolutional networks .

Szeliski, R., 2021. Computer Vision: Algorithms and Applications 2nd Edition. URL: https://szeliski.org/Book.

TensorFlow, 2022, . URL: https://www.tensorflow.org/. accessed: 2022-01-04.

The EUVP dataset, 2021, . URL: http://irvlab.cs.umn.edu/resources/euvp-dataset. accessed: 2022-20-03.

Tripathi, S., Kumar, R., 2019. Image classification using small convolutional neural network, in: 2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence), pp. 483–487. doi:10.1109/CONFLUENCE.2019.8776982.

Umeda, N., Kahn, D., 1981. Frequency of occurrence of word sequences. The Journal of the Acoustical Society of America 70, S41–S41. doi:10.1121/1.2018868.

Vyavahare, A.J., Thool, R.C., 2012. Segmentation using region growing algorithm based on clahe for medical images, in: Fourth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom2012), pp. 182–185. doi:10.1049/cp.2012.2522.

Wang, D., 2019. Single-underwater-image-enhancement-and-color-restoration. URL: https://github.com/wangyanckxx/Single-Underwater-Image-Enhancement-and-Color-Restoration. accessed: 2022-02-03.

Waszak, M., Cardaillac, A., Elvesæter, B., Rødølen, F., Ludvigsen, M., In Press. Semantic segmentation in underwater ship 1 inspections: Benchmark and dataset.

Xiong, J., Zhuang, P., Zhang, Y., 2020. An efficient underwater image enhancement model with extensive beer-lambert law, pp. 893–897. doi:10.1109/ICIP40778.2020.9191131.

Yadav, G., Maheshwari, S., Agarwal, A., 2014. Contrast limited adaptive histogram equalization based enhancement for real time video system, in: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2392–2397. doi:10.1109/ICACCI.2014.6968381.

Yakubovskiy, P., 2019. Segmentation models. URL: https://github.com/qubvel/segmentation_models. accessed: 2022-17-02.

Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y., 2016. Single-image crowd counting via multi-column convolutional neural network, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 589–597. doi:10.1109/CVPR.2016.70.

Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., 2017. Pyramid scene parsing network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2881–2890.

Appendix

A Materal

A.1 Source Code

The source code is provided in the delivered material. Trained weights for the models, hdf5 datasets, png dataset are also provided.

A.2 Videos

Videos of the video segmentation are provided in the delivered material. The videos are also available through the URLs:
Video 1: `https://youtu.be/QsDGcyKQP1Q`
Video 2: `https://youtu.be/xttT9I67Xr8`
Video 3: `https://youtu.be/6b4hjV8SX3g`
Video 4: `https://youtu.be/fvMjTieAvPI`