Master's thesis

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

Håkon Kvalvåg Pettersen

Towards Remaining Life Assessment by use of Conditional Adversarial Networks

Master's thesis in Marine Techonolgy
Supervisor: Sigmund Kyrre Ås
Co-supervisor: Marius Andersen
June 2021

NTNU
Norwegian University of Science and Technology

Håkon Kvalvåg Pettersen

Towards Remaining Life Assessment by use of Conditional Adversarial Networks

Master's thesis in Marine Techonolgy
Supervisor: Sigmund Kyrre Ås
Co-supervisor: Marius Andersen
June 2021

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

NTNU
Norwegian University of
Science and Technology

Preface

This thesis summarizes the work performed in TMR4930 Marine Technology - Master's Thesis at the Norwegian University of Science and Technology (NTNU), and is the final delivery for a Master of Science degree within Marine Structures. The work is a continuation of the project thesis written in the Autumn of 2020, and was carried out from January to June 2021.

The motivation for this thesis was to investigate the use of state-of-the-art machine learning algorithms to assess the fatigue life of mooring chains. The approach evaluates the stress concentrations caused by pitting corrosion by use of a conditional generative adversarial network. The neural network used an image representing the pit geometry as an input and generated an output containing the resulting stress concentration factor at a critical distance.

Similar topic has been investigated in another master thesis by a fellow student, Henrik Heien. Through joint supervision, different approaches were openly discussed. However, this present work has been written in its entirety by Håkon Kvalvåg Pettersen.

It should be mentioned that writing this Master's thesis under the COVID-19 pandemic has been challenging. Guidance meetings were held online throughout the semester, thanks to the supervisors' flexibility, making it easier to seek guidance when local restrictions prevented physical meetings.

The reader is expected to have knowledge of basic fatigue theory and what factors within the marine industry affect the fatigue performance. It is also beneficial to have a basic understanding of machine learning theory. However, relevant machine learning theory will be explained in order to assist the readers to understand the basics concepts and understand the approach of this thesis. A good foundation of linear algebra and calculus is advantageous.

Acknowledgments

I would like to thank my supervisor, Professor Sigmund Kyrre Ås, for excellent guidance throughout the semester. His optimism and deep insight within the field of fatigue have been both inspiring and motivating when working with this thesis. It has been beneficial to be able to hold online meetings and discuss relevant or more casual topics on a regular basis.

Further gratitude will be given to my co-supervisor, Marius Andersen, for providing the input data used in this thesis. Additionally, his participation and engagement in guidance meetings, helpful discussions regarding the thesis and choice of machine learning algorithm have been appreciated.

I would also like to thank my fellow student, Henrik Heien, for open discussions regarding the approach of this thesis and a sharing of his insight within machine learning. A big thanks are given to my colleagues and office mates at Tyholt. The open environment and opportunities for asking for help, discussing, or sharing knowledge even late at night, have undoubtedly been outmost appreciated.

Finally, I would like to thank my friends and family for their unconditional support and motivation throughout my studies at NTNU.

Trondheim, June 10, 2021
Håkon Kvalvåg Pettersen

Abstract

Statistics show numerous mooring lines failure has occurred in early life, well within design life. Mooring chain failure accounted for up to 50 % of total failures, where fatigue was one of the main drivers for failure. To detect and avoid possible failures, inspection by use of, e.g., remotely operated vehicles (ROVs) is commonly done. Combining these inspections with a machine learning algorithm that can process 3D scans can provide fast, efficient, and accurat predictd on the condition of mooring chains without the need for finite element analysis. This would welcome a reduction in operational cost and possibly provide more insight than by today's standard approach.

This thesis presents a method of using a conditional generative adversarial network (cGAN) in remaining life assessment of corroded mooring chains. The constructed neural network is based on the existing image-to-image translation network Pix2Pix. The constructed network translated a gray-scale image of a single pit surface to the corresponding stress field image containing the stress concentration factor (SCF) at a critical distance of $L/2 = 0.5$ mm.

In order to train and test the performance of the neural network, a procedure of creating a finite element model that could obtain the stresses at the critical distance for many different pit geometries was established. This procedure used an artificial sub surface, identical to the pitted surface, to obtain these stresses.

The constructed neural network provided promising results for use in the fatigue assessment of corroded mooring chains. The network's generator was able to generate real-looking stress field images that are similar to the true stress field images. The prediction error in the maximum SCF in the generated images was low and was most likely to be overestimated. By introducing an additional loss function, a maximum value loss, for the generator, the prediction error decreased. However, with the maximum loss, the SCFs were more likely to be underestimated. Hence, the network trained without the additional loss should be used for further fatigue assessments.

Sammendrag

Statistikk viser at fler-tallige tap av forankringsliner har skjedd tidlig i den designede levetiden. Forankringsliner av kjetting var ansvarlige for opptil 50 % av alle tap, hvor utmatting var en av hovedårsakene for disse tapene. For å oppdage og unngå mulig tap, er det vanlig å bruke fjernstyrte undervannsfarekoster (ROV). Ved å kombinere disse inspeksjonene med en maskinlærings algoritme som kan analysere 3D skanninger, kan gi en rask, effektive og sikker prediksjon på tilstanden til kjetting løkker uten behovet for endelige elementmetode analyser. Dette vil redusere kostnader og mulig gi bedre innsikt enn dagens metode tilbyr.

Denne avhandlingen presenterer en metode hvor et betinget opponent-genererende nettverk (cGAN) kan bli brukt i evalueringen av gjenværende levetid for korroderte kjettingløkker. Det etablerte nettverket er basert på det eksisterende bilde-til-bilde transformasjons nettverket Pix2Pix. Det etablert nettverket transformerte ett gråskala bilde av overflaten til én enkel korrosjonsgrop til ett spenningsfelt bilde som representerer spenningskonsentrasjonen (SCF) i en kritisk distanse på $L/2 = 0.5$ mm.

For å både trene og teste ytelsen til det nevrale nettverk, ble det etablert en prosedyre for å lage en endelig elementmodell som kan finne spenningen i den kritiske distansen for mange forskjellige korrosjonsgrop-geometrier. Denne prosedyren brukte en syntetisk overflate plassert under den faktiske overflate. Den syntetiske overflaten ar identisk med den faktisk, og for å oppnå spenningene på den kritiske distansen.

Det etablerte nevrale nettverket ga lovende resultat for bruk i utmattings evalueringen av korroderte kjetting løkker. Nettverkets generator klarte å genere bilder som både ser ekte ut og ligner på de faktiske bildene av spenningskonsentrajon. Feilen i maksimum spenningskonsetrasjon i de generete bilden var lav, og verdien av spenningen ville mest sannsynlig bli overestimert. Denne feilen ble lavere ved å introdusere en ekstra målfunksjon, en maksimum verdi målfunksjon. Med den ekstra målfunksjonen ville spenningen mest sannsynlig bli underestimert. Derfor burde nettverket trent uten den esktra målfunksjonen blir brukt for videre utmattings evalueringer.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence

ANN Artificial Neural Network

cGAN Conditional Generative Adversarial Networks

CNN Convolutional Neural Network

DCGAN Deep Convolutional Generative Adversarial Networks

DL Deep Learning

DNN Deep Neural Network

FE Finite Element

FEA Finite Element Analysis

FFNN Feedforward Neural Network

GAN Generative Adversarial Networks

GPU Graphics Processing Unit

JIP Joint Industry Project

MAE Mean Absolute Error

ML Machine Learning

MLP Multi Layer Perceptrons

MSE Mean Squared Error

NN Neural Network

SCF Stress Concentration Factor

SDG Stochastic Gradient Descent

TCD Theory of Critical Distance

TPU Tensor Processing Unit

Chapter 1

Introduction

1.1 Background and Motivation

Mooring lines are essential for stationary units like FPSOs, FSOs, offshore wind farms, and other floating constructions. To safely be kept stationary at the desired location for a design life of 20-30 years, the mooring lines have to withstand harsh environments and degradation mechanisms such as corrosion and fatigue. Failure of one or more lines could be catastrophic, resulting in oil spills, risk of casualties, and increase costs. Thus, to prevent failures, substantial quality requirements are needed. In terms of fatigue design, a safe life approach is made to sufficiently design the mooring lines against fatigue failure.

Statistics shows that numerous mooring line failures have occurred in early life, well within design life. Over 90% of failures occurred within the first 13 years [1]. Chains, connectors, and wire ropes were the three most common components causing these failures, pointing to pitting corrosion and fatigue as two main drivers. In a study by Fontaine et al. [2] a set 107 mooring line failure was investigated, in which 46% of failure event was associated with chains. Investigations of these chain failures clearly indicated that fatigue and corrosion accounts for 56% of failure events, see Figure 1.1.

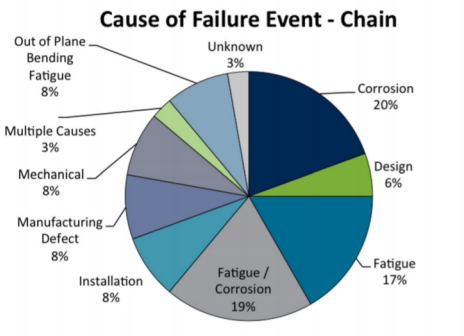


Figure 1.1: Causes of failure event on chains in mooring lines [2]

To better understand the mechanism causing the high rate of failures, especially in mooring chains, joint industry projects (JIPs) have been conducted. Studies on the effects of corrosion on the fatigue life of chains clearly show that pitting corrosion and corrosion in general, reduces fatigue life and thus, increases the probability of failure [3]. Monitoring the condition of the chains requires frequent inspections with remotely operated vehicles (ROVs). Such inspections are costly, takes time, and do not always provide the best pictures of the structural condition. For new industries such as offshore wind, a cost reduction is crucial to be profitable.

Therefore, a solution that provides fast and accurate information is highly welcomed. The use of machine learning has rapidly increased over the last decade. It has proven to provide accurate results from, e.g., image recognition and regression problems. Combining a machine learning algorithm with inspection data in the form of a 3D scan, such as LifeMoor and Karken Robotics Inc. is doing, can ultimately provide accurate predictions on corroded mooring chains' conditions.

1.2 Literature Review

The following section provides an overview of relevant research in the fields of corrosion effects on mooring chains, machine learning in general, and machine learning in structural mechanics. In addition, relevant work within machine learning for the approach of this thesis is described.

1.2.1 Corrosion on Mooring Chains

The integrity of mooring lines is of high relevance for the industry. The topic has been studied in several joint industry projects (JIPs) in order to understand and clarify the mechanisms and impact of conditions affecting the mooring lines. One of such is Mooring Integrity JIP Phase 1 & 2 [4, 5], where the first phase identified the factors that influence the integrity and the challenges for the industry— pointing out corrosion in pitting corrosion alongside fatigue as two critical factors. In

the second phase, further investigations of these factors were conducted. Investigations in Seawater Corrosion of Ropes & Chains (SCORCH) JIP examined pitting corrosion effects on mooring chains [6] and corrosion in mooring lines[7]. Furthermore, experimentally testing the residual strength of severely corroded chains, as seen in Figure 1.2, showed good results[8]. Even with significant material loss due to pitting corrosion, the residual strength was found to be 80-90 % of minimum breaking load, showcasing that the condition is not as bad as first thought. Further, the Finite Element Analysis of Residual Strength (Chain FEARS) JIP [9] was initiated to examine finite element analysis (FEA) performance in assessing the same corroded chains as in the SCORCH JIP.

Figure 1.2: Pitting corrosion on mooring chain[8]

The effect of pitting corrosion on the fatigue life was investigated under the Fatigue of Corroded Chains (FoCCs) JIP. In fatigue assessment of corroded mooring chains, it is crucial to model the chains correctly. Using a uniform thickness reduction may not accurately display the effects of local corrosion on fatigue life. Thus, a study of pitting corrosion effects on fatigue of mooring chains was conducted by Baker et al. [3]. A fatigue test of two pits with different root radius placed at the outer crown and one pit placed at the inner bend was performed to showcase FEA's applicability in assessing fatigue life. In addition, this study demonstrated the performance of simulated damage. The finite element models, FE models, were established using a 3D scan of the chains to create a model close to those tested. However, the surface condition of the chain is rather smooth, forcing the failure at machined pits. The results show that FEA is a reliable method for assessing fatigue life, as the estimated crack initiation point coincides with failure location from the fatigue test. The estimated initiation point was represented by the location of the highest stress concentration factor (SCF). By looking at the local stresses, it was evident that fatigue was dominated by material properties rather than the application of the steel, e.i. the fatigue curve for steel freely corroded in seawater- B1 in DNV-GL RP C203 [10], fit better than the curve for studless chains in DNV OS-E301 [11].

To summarize, the ability to characterize the local stresses in failure locations may enhance the interpretations of the condition of corroded chains.

In terms of fatigue, pit geometry can be considered a geometry defect rather than a chemical process. Based on this, an investigation of finite fatigue life for notched specimens is of interest. Susmel and Taylor [12] came up with a methodology to predict the fatigue lifetime, e.i., finite life, of notched components. The method utilized the linear stress field in close vicinity of the assumed crack initiation point under variable amplitude uniaxial/multiaxial fatigue loading, alongside a Modified Wöhler Curve Method in the plane of maximum variance for the resolved shear stress. The theory of critical distance (TCD) imposed as a material parameter. By use of 124 tests, the methodology showed to be accurate under both constant and variable amplitude loading. It should be noted that the determination of damage sum is the most critical part of the approach and has to be done *a priori*, by, e.g., experiments.

1.2.2 Machine Learning and Structural Mechs

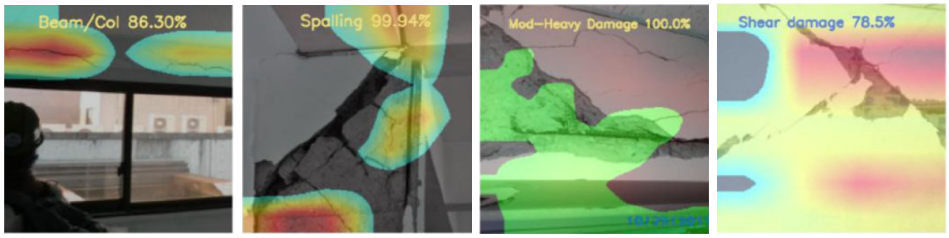
Machine learning (ML) is a powerful tool that can be applied to numerous problems as long as the algorithm is correctly trained. In later years, researches have started to used ML in engineering problems. Combining real-life tests, finite element analysis, and machine learning can significantly reduce the time consumption compared to the standard approach. Another benefit is that ML can find relationships that previously were hard to establish due to the complexity of the problem. In the following, some examples of machine learning used within the field of fatigue and degradation are presented.

Motivating the application of machine learning based on FEA, Ok et al. [13] used an artificial neural network (ANN) to predict the ultimate strength of an unstiffened plate with localized pitting corrosion. From the analysis of more than 256 non-linear finite element models, four parameters were identified as the most weakening; plate slenderness, pit width, length, and depth. These four parameters were therefore used as input to the neural network. With only one hidden layer, the reasonably small network generated an empirical formula for the ultimate strength, which showed good accuracy. Moreover, the number of neurons in the hidden layer had little effect on the accuracy of the network, which indicates that a single-layered network is trained well with a small data set.

In general, obtaining a data set that represents the phenomenons of investigation is crucial. Cottis et al. [14] investigated the reduction of corrosion data. It demonstrates the complexity of modeling the effect of corrosion. As much as ten parameters affect the corrosion behavior, and a small data set compared to the input dimension yields inaccurate predictions. The authors point out that neural networks (NNs) cannot accurately predict conditions far away from the training points. Moreover, there should be a data point in each corner of the hypercube produced by the inputs. In reality, some parameters are more significant than others. Thus a reduction of the input space can be made.

Fathalla et al. [15] used ANN in combination with FEA to predict the remaining life of cracked concrete bridges. The study highlights important aspects of modeling and creating the data set in which neural networks are trained and tested on. The networks can only predict reasonable results for cases it has been trained on. Thus, unrealistic or unexpected crack patterns had to be included in the data set. Somewhat logical phenomenons like symmetric crack patterns, which have the same fatigue life, were not learned by the network itself. Including these effects improved the network's ability to predict the remaining fatigue life. In addition, the procedure presented in the paper is inspiring on how to create sufficient training data and how to find the essential parameters for fatigue analysis.

Image recognition can be used to determine the type of damage or categorize how badly a structure is damaged. Gao and Mosalam [16] used a convolutional neural network (CNN) with transfer learning for binary classification of component type and spalling condition of concrete structures. The network also classified damage level and damage type, with three and four classes, respectively. Figure 1.3 shows a prediction of the trained network with a classification activation map. Using a pre-trained network able to detect low-level features, a limited data set of 2000 pre-labeled images could be used to capture more abstract and high-level features for the damage conditions. By retraining the last two layers of the convolutional layers, the network yielded quite good results.



(a) Component type **(b)** Spalling condition **(c)** Damage level **(d)** Damage type

Figure 1.3: Prediction examples of the Image-Based Structural Damage Recognition network[16]

Regarding mooring line failure, a deep neural network, in the form of a CNN, was used to detect failures[17]. By using the vessel position, an image of the vessel position was created. To include the environmental data, the 6-degree-of-freedom accelerations were added to the images. With these images, the network could classify whether the mooring lines were intact or damage, with high accuracy (over 99 %). The data set was created using numerical models and simulations covering a year of environmental loads. However, the study only used information about the vessel movements and did not include any information about the mooring lines' condition. Thus, it can only predict if lines are damage, act on it after the failure has occurred. To prevent the loss of one or more lines, a neural network that can predict the conditions of the mooring lines is useful.

1.2.3 Image-to-Image Translation

In recent years the use of generative modeling has become popular and proven to provide good results. A variety of approaches have been taken. Some examples are Fully Visablle Belief Network, such as PixelCNN [18], Variational Autoendocers [19], Boltzmann Machines, and Generative Adversarial Network (GAN). The latter has some advantages; it can produce samples in parallel, are asymptotically consistent, and does not require Markov chains, and thus, simplifies the training procedure. GANs were first described by Goodfellow et al. [20] in the paper *Generative Adversarial Networks*. The network comprises of two models, namely a generator, generating samples, and a discriminator trying to distinguish the generated samples from the true sample. These two models are trained in an adversarial game, based and game theory.

As the name suggests, GANs can generate samples such as images. These images are generated without any input. However, for this present thesis, it is wanted that a network can predict or generate samples based on an input sample. Thus, using a conditional GAN is effective. In these networks, the generated sample is conditioned on an input sample. One such network is the Pix2Pix network created by Isola et al. [21] in 2016 and later review in 2018. The network takes an input image, e.g., grayscale image or aerial photo, and generates an output image with color and a map respectively, see Figure 1.4. The network is not limited to finding the mapping from input to output image. It learns loss adapted to the specific tasks, e.i., when translating a grayscale to color, it finds a different loss than that of translating aerial photo to a map. As a positive consequence, the network is applicable for various problems and could be useful for this thesis. The network and its structure will be further explained in section 3.6

(a) Gray scale to color translation **(b)** Aerial photo to map

Figure 1.4: Pix2Pix generated examples[21]

1.3 Objectives

This Master's thesis aims to investigate the use of machine learning in the assessment of remaining life assessment of pitting corroded mooring chains. Today's standard approach is time-consuming and often requires finite element analysis to investigate the reduction in fatigue life due to corrosion thoroughly. Machine

learning, especially with deep neural networks, can drastically decrease time consumption and remove the need for FEA.

Furthermore, the thesis aims to develop a conditional generative adversarial network based on the Pix2Pix network architecture. The network should perform image-to-image translation, which takes the pit-surface elevation as an input image. Based on this input, the desired output of the neural network is a stress field image containing the stress concentration factor at a critical distance of 0.5 mm below the pitted surface.

In order to generate an accurate output image, the network has to be trained on a data set. Thus, the present thesis introduces a method of creating a large data set that contains grayscale images of the stress concentration factors at a critical distance. Lastly, the thesis only describes the applicability for cGANs in fatigue assessment and describes how such a network can be trained to obtain good results. Therefore, the implementation of a neural network as an end-to-end application is not within the scope of this thesis.

1.4 Contributions

The main contributions of this Master's thesis can be summarized as follows:

- Describing a method for creating a data base which contains grayscale images of the stress concentration factors at a critical distance using finite element analysis.
- A method on how to use image-to-image translation in fatigue assessment.
- Create a neural network that accurately can produce a grayscale image of stress concentration below the surface of corrosion pits form mooring chains.

1.5 Thesis Outline

This thesis is organized as follows:

- **Chapter 2** givens an introduction to relevant fatigue theory to understand and motive the choices made during creation of data set.
- **Chapter 3** describes machine learning theory in general and the theory behind GANs and cGANs. In addition, specifics regarding the Pix2Pix network is presented here.
- **Chapter 4** describes the method of creating a data set. The chapter describes how the finite element model was created.
- **Chapter 5** describes the implementation of the neural network, and how it was put together. The training procedure is described.

- **Chapter 6** presents the result from training and testing the neural network. Furthermore, the time consumption of the finite element model is presented.
- **Chapter 7** discusses the results and choice made when creating the FE model.
- **Chapter 8** concludes based on the results and discussion. Lastly, recommendations for further work is given.

Chapter 2

Fatigue

This chapter will present relevant fatigue theory as well as phenomena that affect fatigue performance. It will serve as a foundation for the decisions made throughout the development of a FE model and data collection process. Finally, a method for predicting the reduction in fatigue life is provided to motivate the usage of critical distance in fatigue analysis.

2.1 General Concept of Fatigue

Fatigue damages occur in structural components due to cyclic loading over time. These loads may be well below yield stress. Thus, the damage may not be directly observable, but microscopic damage accumulates in such a manner that the macroscopic cracks occur, and the material loses its ability to carry loads.

Fatigue damage can be divided into three stages, with corresponding number of cycles; the crack initation stage, N_i, the crack growth stages, N_g, and final failure. The total numbers of cycles before failure, e.i. fatigue life, is:

$$N = N_i + N_g \tag{2.1}$$

The crack initiation stage is often difficult to describe, and models describing this stage are often limited to the material. The characteristic of this stage is that the slip bands take place within a few grains at the surface, causing intrusion and extrusion on the surface. Thus, the crack is initiated by slip bands. In the crack growth stage, the growth depends on material properties. It has a higher growth rate compared to the initiation stage [22].

2.2 Stress-Life Approach

When assessing the fatigue life, it is often convenient to look at the stress-life curve, known as the S-N curve. The S-N curve or Wöhler curve was postulated by the German railway engineer August Wöhler, who was among the first to find a relation between the fatigue resistance in a material and the stress amplitude of cyclic stress acting on the material [23].

The S-N curve is established by plotting the stress range, ΔS, against the number of cycles before failure, N. As N may by high, say 10^7, the curve is often plotted in a log-log format, see Figure 2.1.

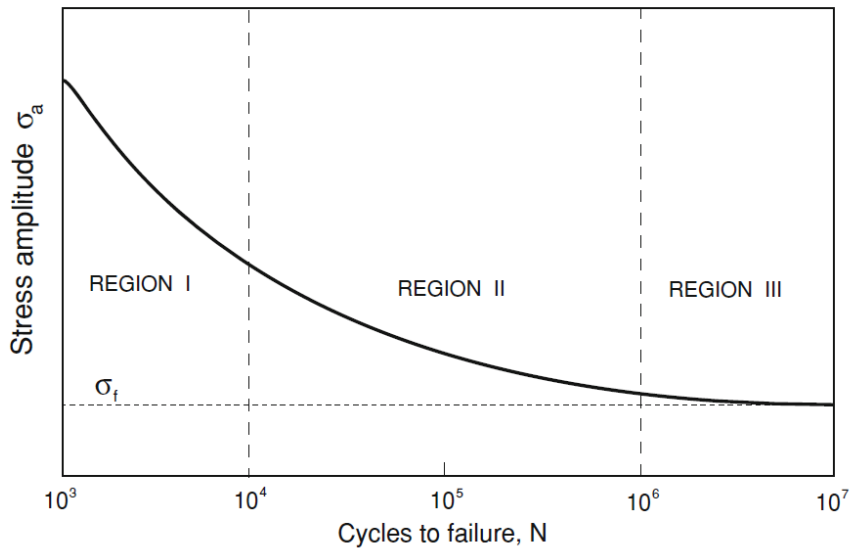


Figure 2.1: Typical S-N curve, stress amplitude versus cycles to failure on log-log format, adapted from [23]. The horizontal and vertical axis shows cycles to failure and the stress amplitude respectively. Region I is the low-cycle region, where as Region II is the high-cycle region. The last region, Region 3 is the fatigue limit region.

In the low-cycle region, **Region I** in Figure 2.1, strain and stress exceed the elastic properties of the material, and Hookes law is no longer valid. Under these stresses and strains, the whole section undergoes macroscopic plastic deformation, to the contrary for high-cycle fatigue where only the surface undergoes deformation. Consequently, the fatigue crack grows along grain boundaries and eventually becomes a continuous macro crack. When the crack becomes large enough, it is driven by the external load and will continue to grow in a plane normal to the load [23].

Region II is called the high-cycle region. The material is expected to behave elastically on the macroscopic level, and S-N data usually follows a log-linear relation,

also known as the Wöhlers equation.

$$N(\Delta S)^m = A \tag{2.2}$$

On log-log form:

$$\log \Delta S = -\frac{1}{m} \log N + \frac{1}{m} \log A \tag{2.3}$$

where A and m are constants depending on the material. The starting point for high-cycle region varies from 10^2 to 10^4 depending on the material [24]. For marine structures, e.g. mooring chains, a long life-time is expected, meaning that number of cycles often lies within **Region II**.

For ferrous metals, the curve flattens out for low stress ranges, seen in **Region III**. Below a stress range ΔS_0 no fatigue damage will occur due to the formation of non-propagating cracks. This is known as the *fatigue limit*. However, for nonferrous metals and corrosive environments, small pits can be present and initiate the crack growth. Thus, the fatigue limit does not exist, and fatigue damage occurs at all stress levels [22].

2.2.1 Effects on the S-N Curve

High-cycle fatigue is often associated with a stress-based approach. It focuses on the stresses in the affected region of the component. Apart from the stress range, mean stress, environmental effect such as corrosion, and geometrical effects, e.g., notches, are important parameters that influence the fatigue life. The latter will be discussed in section 2.3. Furthermore, it should be emphasized that effects such as shape and distribution of inclusion, surface finish, grain size and direction, component size, load type, surface treatments, and temperate will affect the stress-life. The reader is referred to chapter 3 in the Fatigue and Corrosion in Metals textbook [23] further details.

Mean-Stress Effect

For an engineering component under uniaxial cyclic loading history following a sinusoidal pattern, as shown in Figure 2.2, the following relations can be established.

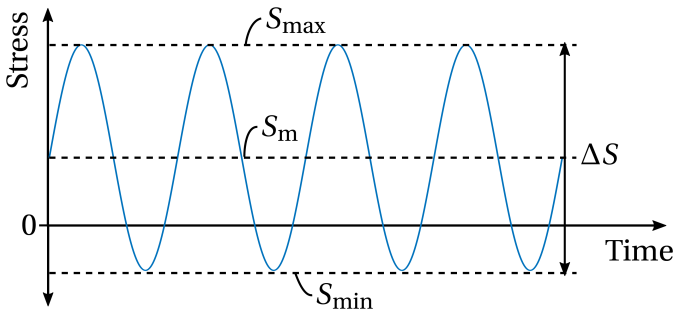


Figure 2.2: Cyclic stress history and symbols [22]

The stress range:

$$\Delta S = S_{max} - S_{min} \tag{2.4}$$

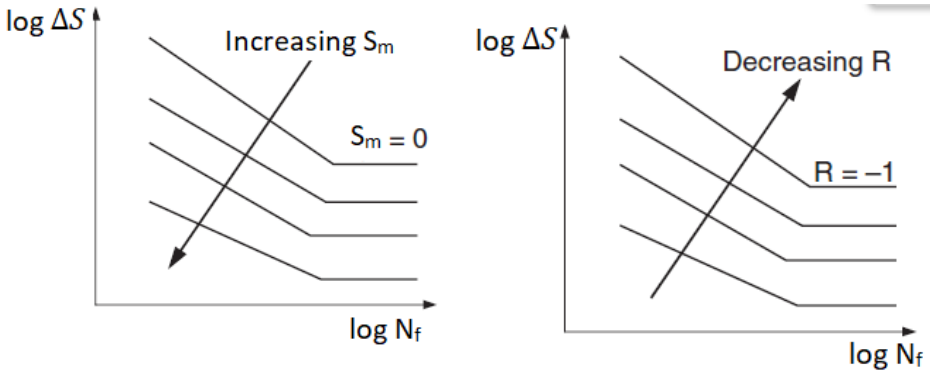
The stress ratio:

$$R = \frac{S_{max}}{S_{min}} \tag{2.5}$$

The mean stress:

$$S_m = \frac{\Delta S}{2}\left(\frac{R+1}{R-1}\right) \tag{2.6}$$

An increase in the mean stress, S_m, yields increasingly fatigue damage. This can be seen in Figure 2.3a, where the curve moves downwards as S_m increases. A higher mean stress implies most of the stress range is tensile stress, i.e., contributes to crack growth. On the other hand, as the load ratio, R, decreases, the fatigue damage abates, moving the fatigue limit upwards, as seen in Figure 2.3b. A lower stress ratio implies that the cyclic stress on the tensile side becomes smaller, and hence, lowers the fatigue damage as only tensile stresses contribute to crack growth.



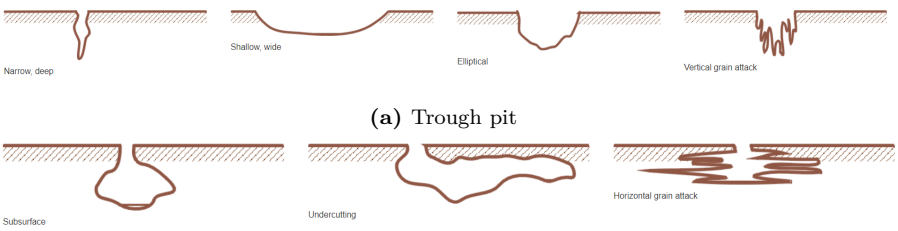
(a) Increase in S_m **(b)** Decrease in R

Figure 2.3: Effects of mean stress and stress ratio [25]

Environmental Effects

For mooring chains, a sudden unexpected failure due to fatigue may occur. This may occur even if the experienced number of cycles is well below the designed fatigue life or even the stress range is lower than the fatigue limit (if such exists). Hostile environments, e.g., seawater, can accelerate the crack initiation and growth, due to formation of corrosion pits acting as stress raisers. Additionally, chemical reactions and dissolution at the crack tip cause the crack to grow faster and reduce fatigue life. In general, fatigue limits do not exist for corrosive environments, and fatigue damage occurs at all stress levels. This emphasis the importance of correctly calculate the stresses within components such as mooring chains to ensure a safe fatigue life design or correctly predict the condition of the chains.

For mooring chains, pitting corrosion is one environmental effect that reduces the fatigue life, as these pits act as stress raisers. Pitting corrosion is localized corrosion on the surface in which cavities or holes are created. The pits are formed when the corrosion attacks a point or a small area. The geometry of the pits varies and will, depending on the shape, influence the fatigue performance differently. In Figure 2.4 a variety of their cross-sectional shape is shown.



(a) Trough pit

(b) Narrow pit

Figure 2.4: Different cross-section geometries of pitting corrosion[26]

2.3 Notch effect

The stress amplification naturally depends on the geometry of the notch, in which depth and root radius are the most weakening parameters. The theoretical stress concentration factor, k_t, is based on linear elasticity theory. It is determined by the ratio between the peak stress at the notch tip, σ_{max}, and the nominal stress, σ_{nom}, based on the net section, i.e., stresses not affected by the stress concentration. See Equation 2.7.

$$k_t = \frac{\sigma_{max}}{\sigma_{nom}} \tag{2.7}$$

Due to the local stress increase, it is expected that the fatigue limit will be reduced. Evidence shows that the reduction is present but not as high as k_t. Thus, the actual stress concentration factor, called the notch factor, k_f, is equal to or lower than k_t. When the latter is high, k_f tends to deviate more from k_t. In addition, for hard materials, SCF and notch factor tends to become closer [23].

2.3.1 Petersons and Neubers Approach

Peterson introduced a notch sensitivity factor, q, showing the influence of SCF on the notch factor. The sensitivity factor takes material properties and geometrical effects into account. The notch factor is found by the following equation:

$$k_f = 1 + q \cdot (k_t - 1) \tag{2.8}$$

where $q \in [0, 1]$. When $q = 1$, the notch factor equals the SCF. As k_t is greater than or equal to k_f, choosing $q = 1$ is the most conservative choice. When q equals zero the notch effect is abscent.

Peterson's approach, known as the point method, states that fatigue damage can be determined by the stress level at a short distance below the notch root. In this region, it is assumed that the stress gradient is steep, and there is a linear decrease in notch tip stress. Through experimental work, Peterson came up with the empirical expression for q as seen below.

$$q = \frac{1}{1 + \sqrt{\frac{a_P}{\rho}}} \tag{2.9}$$

Here ρ represents the root radius, and a_P is a critical distance depending on the material.

Similarly, Neuber's approach, which actually was presented before Petersons, is based on stress levels close to the root of the notch. Stresses in close vicinity of the notch are averaged out over a material distance. Almost the same equation emerges.

$$q = \frac{1}{1 + \sqrt{\frac{a_N}{\rho}}} \tag{2.10}$$

The main difference in these two equations is the material constants, a_N and a_P, and how they are derived. Both, Neuber's and Peterson's approach highlight important effects of notches which can be summarized as follows:

- In addition to the peak stress, the stress gradient affects the fatigue strength.
- Root radius influences the strength. It means that similar discontinuities with the same k_t, but with different root radius, will influence differently.

2.4 Critical Distance

Both Neuber's and Peterson's approach is a starting point for the theory of critical distance (TCD). However, at that time, describing the linear stress fields in close vicinity of the notch tip was no easy task and could not practically be done. Nowadays, thanks to tools as the finite element method, a finite element analysis using computers can be used to establish the stress field.

The critical distance, L, is a material parameter. It is found by combining the threshold stress intensity factor, ΔK_{th}, with the fatigue limit range for plain specimens, ΔS_0. See Equation 2.11. If the stress level at a distance $L/2$ from the notch tip is below a critical stress level, no fatigue damage will occur. Thus, it acts as a threshold limit. This method is known as the point method.

$$L = \frac{1}{\pi}\left(\frac{\Delta K_{\mathrm{th}}}{\Delta S_0}\right)^2 \tag{2.11}$$

Other TCD formulations using the critical distance is the line method, in which the stresses are averaged out over a length of $2L$, as illustrated in Figure 2.5c. The

area method, which averaging the stresses over a semicircular area with radius L, is seen in Figure 2.5d.

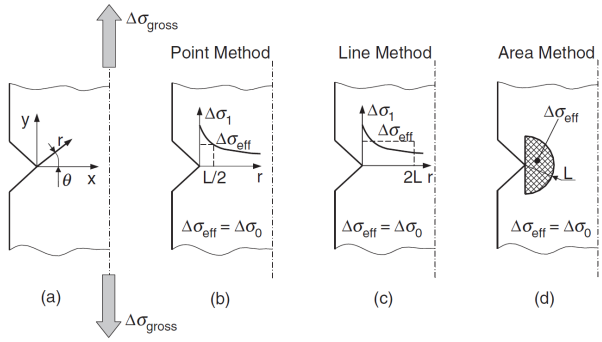


Figure 2.5: Different approaches for theory of critical distance [25]

2.5 Notch Effect on S-N curve

The notch effect influences the whole S-N curve, not only the fatigue limit. To motivate the use of TCD, this present section presents a method, based on section 2.7.2 in [25], where the critical distance is used to calculate the reduction in fatigue life on notch specimens.

The theory of critical distance is used as a starting point when calculating the notch effect. The critical distance, L_M, changes when the number of cycles before failure, N_f decreases. The relation shows that L_M is a function of N_f, which follows the power-law shown below. A and B are constants whom must be determined by experiments.

$$L_m(N_f) = A \cdot N_f^B \tag{2.12}$$

Susmel and Taylor [27] proposed a procedure to determine the constant. The procedure applies two calibration curves as shown in Figure 2.6. At specified value $N_f = N_{f,k}$ there is a corresponding stress range, in the figure shown as $\Delta\sigma_{1,k}$ for plain specimen. For a notched specimen, it is possible to find a distance, $L_m(N_{f,k})/2$, from the notch in which the stress range is the same as the one applied for the plain specimen. By calculation critical distances in both the low- and high-cycle regions, A and B can be determined[23].

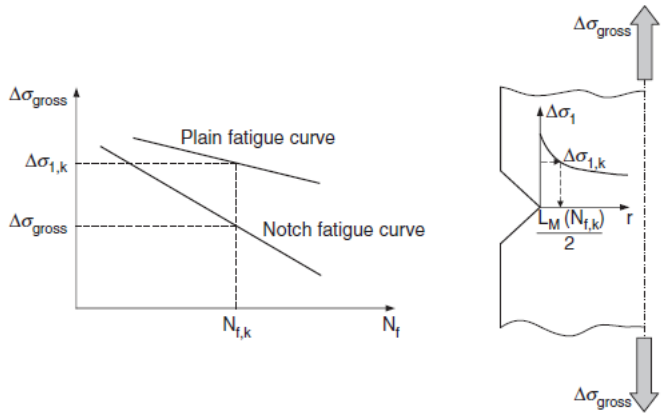


Figure 2.6: S-N curve used for determination of critical distance, and thereby used for determine A and B [25].

A recursive procedure can be used to find the reduced number of cycles before failure, $N_{f,b}$. For a notched specimen with a nominal stress range, ΔS, it is assumed a value for the number of cycles before failure, $N_{f,a}$. By equation (2.12) the corresponding critical distance is calculated. The stress range at a distance $L_m/2$, namely $\Delta S_{1,a}$, can be found using the stress field around the notch tip. From the Wöhler equation, Equation 2.2, presented earlier, the following relation can be obtained to calculate the reduced number of cycles.

$$N_{f,b} = N_0 \left(\frac{\Delta S_0}{\Delta S_{1,a}} \right)^m \tag{2.13}$$

Here ΔS_0 and N_0 represents the fatigue limit stress and the numbers of cycles at the the point of fatigue limit, respectively. If $N_{f,a} \neq N_{f,b}$, the procedure is redone with replacing the assumed number of cycles with the one calculated, i.e.$N_{f,a} = N_{f,b}$. The procedure is repeated until the two coincide.

Chapter 3

Deep Generative Modeling

This chapter explains machine learning in general, and how generative adversarial network is built and trained. The use of deep learning, specifically deep neural networks (DNN), as function approximators and their generalization capability will be discussed. Moreover, image-to-image translation by use of a conditional GAN named Pix2Pix created by Isola et al. [21] is discussed.

3.1 Machine Learning - Introduction

Artificial intelligence (AI) has been present since the 1950s. The concept of AI is that any application that tries to replicate human behavior either by solving a specific task or *learn* how to solve them is artificial intelligence. The latter is Machine Learning (ML), a sub-class of AI. An ML algorithm is not specifically programmed to solve the problem; it rather learns how to solve the problems by recognizing patterns from large amounts of data. Due to the advance in computer processing capacity and a large amount of available data, ML has become faster and cheaper and thus more used. An ML technique is Deep Learning (DL), which is a multi-layered neural network inspired by the human brain. Figure 3.1 summarizes the classes of AI, ML and DL.

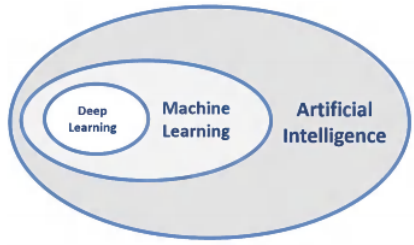


Figure 3.1: Visual representation of classes AI, ML and DL [28]

3.1.1 Supervised vs. Unsupervised Learning

Machine learning can be divided into two categories, supervised learning and unsupervised learning. Supervised learning is when an algorithm has an input variable, x, and an output label, y. The algorithm is trained by showing the input, predicting the output, and comparing the output with the label. This is schematically shown Figure 3.2a. The model is corrected to make the predicted output become more like the label, i.e., finding the best mapping from input to output. Therefore, supervised learning is often used in classification tasks, object detection, regression problems.

(a) Supervised learning **(b)** Unsupervised learning

Figure 3.2: Examples of Machine Learning approaches [29]

In addition, there is unsupervised learning. In this case, the model is trained with a data set that does not contain any labeled data. There exist no output labels, y, see Figure 3.2b. Unsupervised learning models focus on finding patterns or extracting information from the input data instead of predicting an output sample. Such models are able to generate samples based on the distribution found in the data set. Generative models, e.g., Generative Adversarial Networks, are based on unsupervised learning.

3.2 Neural Networks

3.2.1 A Single Neuron - The Building Block

The building block in artificial neural network (ANN), or neural network (NN), is a neuron, also known as a node, a representation is shown Figure 3.3.

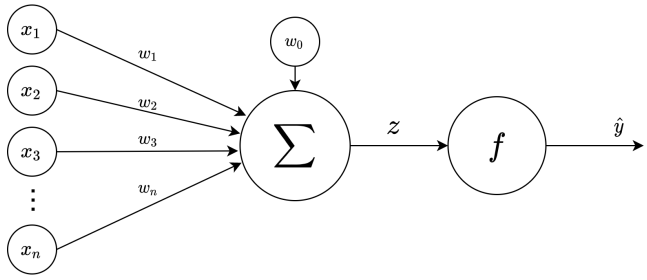


Figure 3.3: Representation of a single neuron used in neural networks

The neuron has inputs $x_1, x_2, ..., x_m$ with corresponding weights $w_1, w_2, ..., w_m$. The weights describe each inputs importance on the the output of the neuron. In addition, a bias, w_0, is connected to the neuron. The bias is meant to introduce a shift to the neuron's output. A summation of the inputs with weights and the bias produces a scalar

$$z = \sum_{i=1}^{n} x_i w_i + w_0 = \mathbf{w}^T \mathbf{x} + w_0 \tag{3.1}$$

where $\mathbf{x}$ and $\mathbf{w}$ is the input and weight vector, respectively. The scalar is feed into an activation function$f$, also known as a transfer function, and yields the output $\hat{y}$.

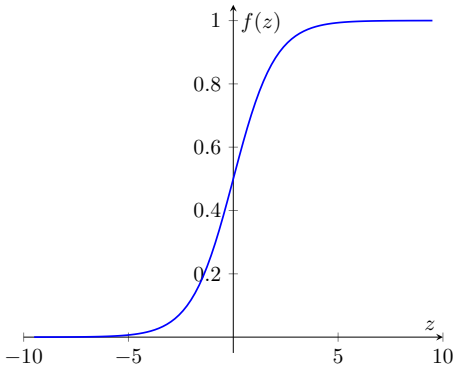
$$f\left(\mathbf{w^T x} + w_0\right) = f(z) = \hat{y} \tag{3.2}$$

3.2.2 Activation Functions

The activation function introduces non-linearity to the network, which enables it to predict non-linear problems. Without an activation function, the output will only be a linear transformation of the input [30]. The choice of activation function depends on the properties of the data and what kind of problem the network is intended to solve. In the following, the most common functions will be presented.

Sigmoid

The Sigmoid function, plotted in Figure 3.4, is a logistic function which squishes value between zero and one. Thus, it fits for yielding the probability as a result. In addition, the derivative exists in all point which enables training with the use of gradient descent. This will be further explained later. One possible problem is that the gradient becomes small for both large and small numbers, thus making training slower.



$$f(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

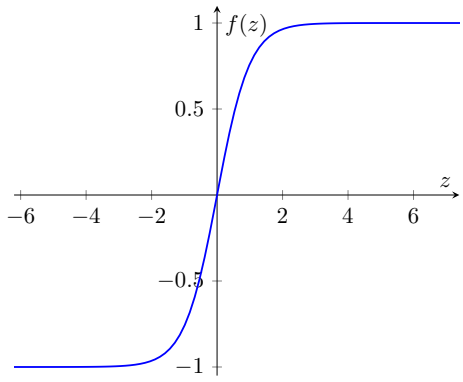
Figure 3.4: Sigmoid activation function

Softmax

The softmax activation function is a generalized Sigmoid function. It transforms an n-dimensional vector into an n-dimensional output vector with a length equal to one. Hence, each component represents the probability of the n elements. The activation function is often seen in the last layer of networks [31].

Hyperbolic Tangent

Hyperbolic tangent, tanh, activation function is similar to the Sigmoid function. The main difference is that tanh is zero-centered and outputs values in the range $[-1, 1]$. As a result, hyperbolic tangent activation is often chosen instead of a Sigmoid activation as negative inputs will be mapped to negative outputs, making it better for training [30].

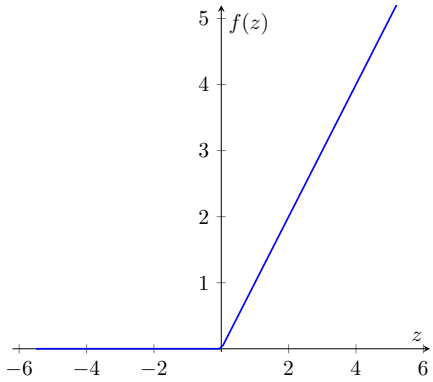


$$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.4}$$

Figure 3.5: Hyperbolic tangent, tanh, activation function

Rectified Linear Unit - ReLU

The rectified linear unit, ReLU, maps the input values in the range $[0, \infty]$. The function is zero for all negative values and acts as a linear function for positive inputs. ReLU is the most common activation function due to being computationally efficient. However, the derivative is constant equal to zero for negative values, which may impose problems during training as neurons can *die* out [32].

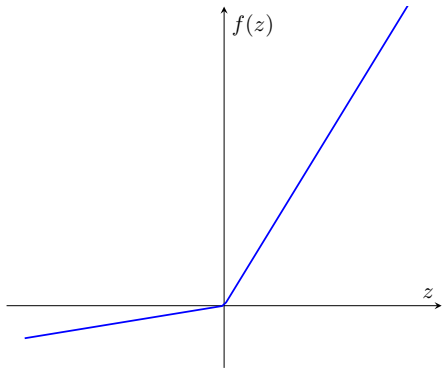


$$f(z) = max(0, z) \tag{3.5}$$

Figure 3.6: Rectified Linear Unit, ReLU, activation function

Leaky Rectified Linear Unit - Leaky ReLU

Leaky ReLU is a variant of ReLU that addresses the problem with a constant zero derivative. For negative values, the function now has a small slope, a, greater than zero.



$$f(z) = \begin{cases} z & z > 0 \\ az & z < 0 \end{cases} \tag{3.6}$$

Figure 3.7: Leaky Rectified Linear Unit, Leaky ReLU, activation function

3.2.3 Deep Feedforward Network

A deep feedforward network, also known as feedforward neural networks (FFNN) or multi layered perceptrons (MLP), is, in essence, connecting multiple neurons together in numerous layers. Figure 3.8 shows a fully connected feedforward network, meaning that all neurons are connected to the neurons in the previous layer. This is often referred to dense layers or densely connected layers. Moreover, the information flows from the input layer to the output layer, i.e., it propagates forward, thereby its name.

Between the input and output layer lie the hidden layers. The hidden layers capture more abstract and complex features from the input data as the width (number of neurons) and depth (number of hidden layers) increases [33]. The state of the hidden layers is not directly determined by the real world. It is also not directly observable, hence given the name hidden layers.

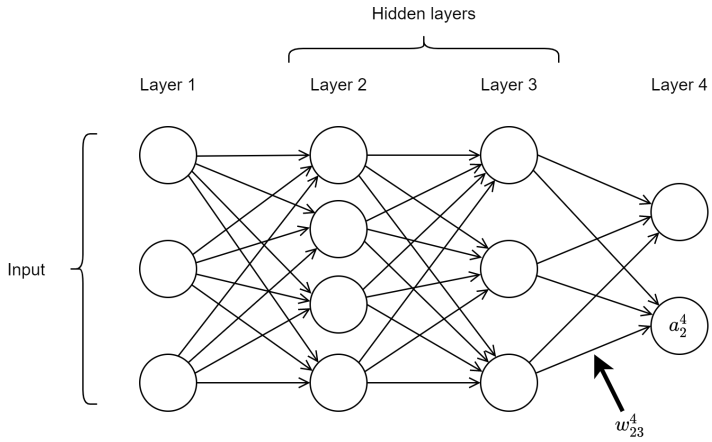


Figure 3.8: A dense neural network with two hidden layers

Using the building block from the previous section, the neuron, it is clear that the output from the first hidden layer now becomes a vector dependent on the input layer. This can be further generalized; let L be numbers of layers and l represent a specific layer. Within layer l, there exist K_l neurons, where k refers to a specific neuron. Furthermore, let w_{jk}^l denote the weight connection between neuron k in layer $(l-1)$ to neuron j in layer l, and w_{0j}^l is the bias. Thus, the output a_k^l of neuron k in layer l, is dependent on the outputs and weights in the previous layer, $(l-1)$, and can be written as seen below.

$$a_j^l = f\left(\sum_k^{K_{l-1}} w_{jk}^l a_k^{l-1} + w_{0j}\right) \tag{3.7}$$

To handle large networks it can be convenient to write Equation 3.7 on matrix

form, as seen in Equation 3.8. Thus, the outputs form layer l is now a vector, $\boldsymbol{a}^l$, where $\mathbf{W}^l$ is the weight matrix and $\boldsymbol{w}_0^l$ is the bias vector.

$$\boldsymbol{a}^l = f\left(\mathbf{W}^l \boldsymbol{a}^{l-1} + \boldsymbol{w}_0^l\right) \tag{3.8}$$

Function Approximators

MLPs are so-called universal function approximators. Cybenko [34] showed that a neural network with just one hidden layer including a finite number of neurons can approximate any continuous function on a closed bounded subset of $\mathbb{R}^n$ by use of a non-linear function as the Sigmoid function [35]. Later, Hornik [36] showed that application of any bounded non-constant activation function, MLPs are universal approximators. In addition, with a sufficient smooth activation function, the derivatives are approximated accurately.

Even though a network with a single hidden layer is sufficient, a deep neural network is commonly used. A reason for this is that the complexity of this single layer may become large, and the width grows exponentially, i.e., numbers of neurons become huge. Therefore, it can be difficult to train the network in such a way that the correct weights are found. Also, for a complex network, there is a risk of overfitting. Nevertheless, based on empirical observations, DNNs have shown to outperform a single hidden layer network although they have the same approximation ability [32].

3.2.4 Optimizing the network

The network has to be trained in order to find the correct weights and biases. Under supervised learning, the network is given both the input and the true output. In order for the network to know how good or bad the prediction is, the quality of the prediction has to be quantified. This is done by using an objective function, also called a loss function. A well performing network has a small error. Hence, the objective when training a network is to minimize the loss function. The process of minimizing the loss is done through backpropagation.

Another essential aspect of optimizing is that the data is split into a training set and a test set. Then, the network is trained on the training data and tested on unseen data from the test set. In this way, it can be shown that the network predicts the output in a suitable way rather than memorizing each data point in the data set. Furthermore, this means that the weights and biases are tuned correctly.

Loss function

The loss function, $\mathcal{L}$, quantifies the error. It can be formulated in a variety of ways, depending on the task at hand. For regression problems, e.i. the network is going to predict numbers, the mean square error (MSE), see Equation 3.9, or mean

absolute error (MAE), see Equation 3.10, is common choices.

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \tag{3.9}$$

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i - \hat{\mathbf{y}}_i \tag{3.10}$$

Here n denotes the number of training samples, $\mathbf{y_i}$ is the true value, and $\hat{\mathbf{y}}$ is the network's prediction.

For other problems, such as classification, where the network predicts a probability of the outputs, the cross-entropy loss function, also known as the log-loss, is commonly used. The cross-entropy, H, of probability distributions p relative to the probability distribution q with n discrete states is given in Equation 3.11.

$$H(p, q) = \mathbb{E}_p\left[\log q\right] = -\sum_{i=1}^{n} q\left(y_i\right)\log\left(p\left(y_i\right)\right) \tag{3.11}$$

Backpropagation

During training, the objective is to minimize the loss function. This can be done by using the gradient of the loss function with respect to the weights, i.e., $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$. Note that the bias is considered as the zeroth weight and that $\mathbf{W}$ represents all the weights in the network. By moving a small step, η, in the direction of steepest descent, the new weights can be found as shown in Equation 3.12. The value η is called the learning rate and prevents overshooting. If the overall objective is to maximize the loss function, one moves in the direction with the steepest ascent, flipping the sign on the gradient below.

$$\mathbf{W} = \mathbf{W} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \tag{3.12}$$

Finding the gradient of $\mathcal{L}$ is, in essence, applying the chain rule from calculus repetitively. This implies that the derivative in each layer is multiplied together, and thus, explains the name backpropagation as the derivative propagates backward in the network. The reader is referred to chapter two of *Neural Networks and Deep Learning* by Nielsen [37] for further details on backpropagation. Nonetheless, in reality, calculating the gradient is time-consuming and is not suitable in neural networks as the training time increases. Therefore, other methods like stochastic gradient descent (SGD) [38], which updates the weights for every training example, Mini-Batch Gradient descent, or adaptive learning rates, i.e., changing η during training, can be used to lower training times.

In recent years, more complicated optimization algorithm has been made. Momentum can be implemented, which can be imagined as pushing a ball downhill. The ball will move faster when the slope is in the same direction as the ball was

initially moving, i.e., the momentum increases in the dimensions in which the gradient points the same way. On the other hand, the ball will move slower when the slope is in other directions; thus, the momentum decreases in the dimensions in which the gradient changes [39]. In addition, adaptive learning rates can be used. The algorithm *Adam* [40] is one that implements both an adaptive learning rate and a property similar to the momentum, with momentum parameters β_1 and β_2 [39].

3.2.5 Generalization

One powerful aspect of NNs is their generalization ability. It means that a sufficiently trained network can predict the output from yet unseen inputs. In other words, both the training error and the test error are small. However, it is not guaranteed that a network is generalized. During training, the network can either be under fitted, generalized, or overfitted, see Figure 3.9.

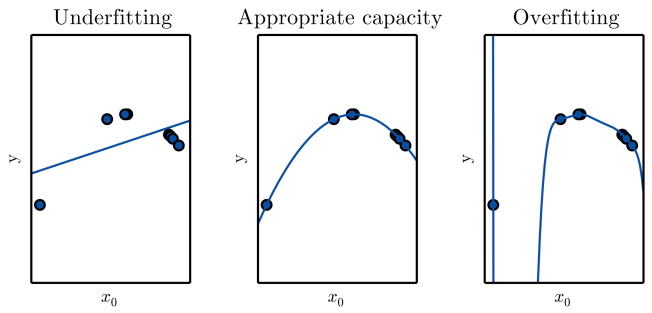


Figure 3.9: Representation of underfitting, generalization and overfitting [35]. The leftmost image is underfitting, where the predicted line is not a good fit to the data points. The middle image shows a generalized network, where the predicted line is a good fit. The rightmost image shows overfitting. Even the the predicted line goes trough all the data points, it do not capture the underlying trend in the data set.

Underfitted means that the network is not performing well enough; both the training and the test errors are too high. This implies that the network is not appropriately trained to find the correct weights and biases due to either a small training data set or that the training data is not a good representation of the whole problem. It can be avoided by increasing the network's complexity, remove noise data, or making sure that the data represent the problem.

An overfitted network remembers each data point rather than capture the whole picture. An overfitted network captures the noisy data. A cause of overfitting is that the network is trained with too much data so that the network learns each point. One can either reduce the size of the data set or decrease the complexity of the network if applicable. Another way to check if a network is overfitted is by looking at the training and test error. If the training errors decay while the test error increases, the network is overfitted.

3.3 Convolutional Neural Networks

A convolutional neural network (CNN) is a distinct type of DNN. CNNs are widely used within image recognition, image classification, and object detection. The reason for its name is due to the convolution layer. A typical CNN architecture can be seen in Figure 3.10. The main layers are the convolutional layer, pooling layer, and often a softmax layer.

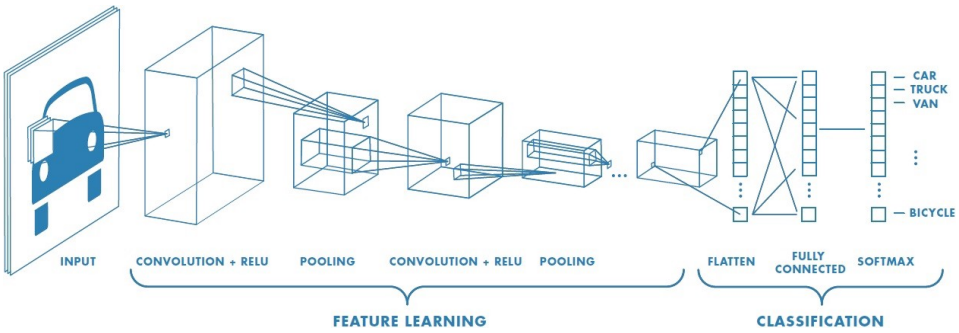


Figure 3.10: Typical layout of a CNN [41]

Convolution layer

A convolution layer aims to extract different features from the input data, often considered as an image. The layer can find features such as edges or patterns, or sharpen the picture or blur it up. It works by convolving a kernel over an image, as shown Figure 3.11. The kernel, also known as a filter, is swept across the image, and the output, which is saved in a feature map, is essentially the dot product of the picture and the kernel as it goes over the image. The filter is moving s pixels every time, these moves are known as strides. A convolutional layer is often seen with a ReLU or LeakyReLU activation function. In Figure 3.11 a 3×3 filter, $\mathbf{K}$, with 1×1 stride is convolved with the input image, $\mathbf{I}$.

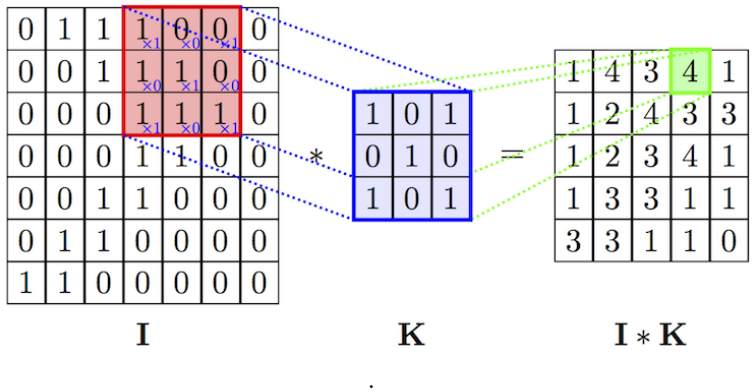


Figure 3.11: Representation of a convolutional layer [42]. The kernel, **K** is moved over the image, **I**, and the dot product of them is the result of the convolutional layer

3.4 Generative Adversarial Networks

Generative adversarial networks are generative models commonly based on deep learning. The GAN architecture and procedure for training were first described in the paper *Generative Adversarial Networks* by Goodfellow et al. [20]. However, the established model was both hard to train and unstable. Thus, the standard approach today is loosely based on the Deep Convolutional Generative Adversarial Network (DCGAN) established by Radford et al. [43].

The architecture of GANs comprises of two models; a **generator** to generate new samples, and a **discriminator** that tries to classify whether samples are fake or not, see Figure 3.12. In a game theory-based procedure, these two networks are pitted against each other. The generator seeks to deceive its adversary, the discriminator, while the generator tries to correctly categorize the generated sample. In this setup, both the generator and discriminator can be optimized to perform better. In the following sections the generator, discriminator and how to optimize the network will be explained.

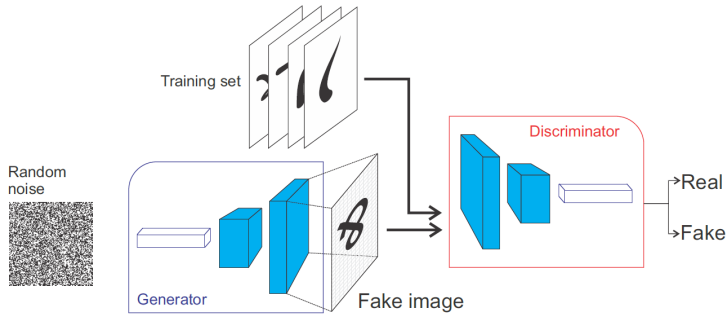


Figure 3.12: Schematics of the architecture of GANs [44]. The generator is feed with random noize vector, and then generates an image. The discriminator assigns a real or a fake label on the input image, either from the training set of the generated image.

3.4.1 Generator

The generator is a differentiable function, G, often represented in the form of a deep neural network, as seen in Figure 3.13 where deconvolution layers are used. The overall goal for the generator is to find a distribution, p_g with θ_g variables, that is as close as possible to the actual distribution of the data set, p_{data}. To learn p_g, the generator is given input z, drawn randomly from a Gaussian distribution, and used as a seed for the generator to create new samples. The input vector is referred to as a latent variable, meaning that it has no physical meaning and is not directly observed. However, it is vital for the network to generate samples.

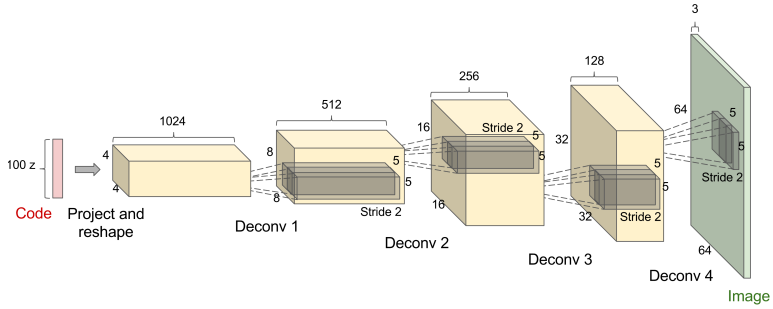


Figure 3.13: Overview of a typical generator architecture [45]. The latent variable, z, is used as input, and decovolutional layers followed by a activation function are used to generate an image.

3.4.2 Discriminator

The discriminator is a differentiable function, D, often as a classification neural network. The discriminator, consisting of θ_d parameters, takes an input sample $\boldsymbol{x}$,

and outputs a single scalar. Thus, $D(\boldsymbol{x};\theta_d)$ represents the probability of $\boldsymbol{x}$ being drawn from the true data set, p_{data}. After training, the need for the discriminator is not there, as only the generated samples are of interest. Thus, the discriminator is only used for training, i.e., making the generator better.

3.4.3 Training Process

A generator that performs well generates samples that fools the discriminator, i.e., maximizing the probability of being real. However, on the other hand, a discriminator that performs well can easily distinguish between a real or fake sample. Thus, it is a two-player game, where the two models try to outperform each other.

More formally, D is trained to maximize the probability of assigning the correct label (real or fake) to both the generated sample and samples from the training set. Further, G is trained to minimize the probability of being classified as fake, i.e, $\log(1 - D(G(z)))$, or the equivalent maximizing the probability of being real $\log(D(G(z)))$.

The standard discriminator loss function, $\mathcal{L}_D$, is formulated as seen in Equation 3.13. Here $\mathbb{E}_{\boldsymbol{x}\sim \mathrm{pdata}}[\log(D(x)]$ represents the expected value of the cross-entropy loss between $\boldsymbol{x}$, drawn from p_{data}, and $D(\boldsymbol{x})$. In some cases, the $\frac{1}{2}$ is discarded in training, as the actual value of the loss function is not of interest, only the optimized point. In fact, the one-half factor shows that the discriminator is trained with two mini-batches; one drawn from training data and one drawn from the generator samples.

$$\mathcal{L}_D(D,G) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p\mathrm{data}}[\log D(\boldsymbol{x})] - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}}[\log(1 - D(G(z)))] \tag{3.13}$$

The generator loss, $\mathcal{L}_G$, is determined by how the two-player game is set up. A well-known way is to set up as a zero-sum game or minimax game. In this game, the generator loss is defined as seen in Equation 3.14, where the generator tries to minimize this loss.

$$\mathcal{L}_G(D,G) = \mathbb{E}_{\boldsymbol{z}}[\log(1 - D(G(z)))] \tag{3.14}$$

Moreover, as $\mathcal{L}_D$ is dependent on $\mathcal{L}_G$, the game can be summarized with a value function, V, where both the parameters for the generator and discriminator, θ_g and θ_d respectively, are included, see Equation 3.15. Thus, both models can be trained based on this function.

$$V(D,G) = -\mathcal{L}_D \tag{3.15}$$

The minimax game originates from that the inner loop is maximized, while the outer loop is minimized, as specified below.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p\mathrm{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}}[\log(1 - D(G(z)))] \tag{3.16}$$

The two-player game is implemented using an iterative, numerical approach. If the inner loop, i.e., maximizing the discriminator, is trained to completion, the network would be overfitted; the discriminator would remember all the data points in the data set. Moreover, in this instance, the discriminator would always assign the correct label to the generated samples and thus prevent a good training of the generator. Therefore, the approach optimizes the discriminator with k steps, followed with one step of optimizing the generator. The optimizing procedure follows **Algorithm 1** presented in paper *Generative Adversarial Networks* [20]. This algorithm is reproduced below.

Algorithm 1: Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. The authors in [20] used k = 1, the least expensive option, in the experiments.

for *number of training iterations* **do**
  for *k steps* **do**
    - Sample a minibatch of m noize samples $\left\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\right\}$
    - Sample a minibatch of m generated samples $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\right\}$
    - Update the discriminator by ascending it stochastic gradient:
      $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$
  end
  - Sample a minibatch of m noize samples $\left\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\right\}$
  - Update the discriminator by ascending it stochastic gradient:
    $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$
end

Furthermore, when using the minimax game, the generator loss may not provide gradients on which the generator can train on. That is, if the discriminator performs well, it can correctly classify the generated sample. As a consequence, the value of $\log(1 - D(G(z)))$ becomes small, and thus, the gradients for the generator to train on becomes small. To prevent this saturated game position, the generator can be trained to maximize $\log(D(G(z)))$, which provides sufficient gradients early in training. Note that in this particular case, the two-player game is not a minimax game. However, the loss functions can find the same optimized point in the adversarial game [20].

3.5 Conditional Generative Adversarial Networks

A limitation to GANs is that they are only able to generate random samples, the samples may be different, but the user does not have any control over the output. An extension to GANs, in which the user can affect the output, are conditional generative adversarial networks developed by Mirza and Osindero [46]. The network has the two same building blocks as GAN, namely a generator and a discriminator. However, both the generator and discriminator are now conditioned on an additional input, $\boldsymbol{y}$, represented in Figure 3.14. This condition can be viewed as telling the network what to generate, e.g., telling the network to generate an image of a cat, or using an image to tell the generator to mimic the style the input image.

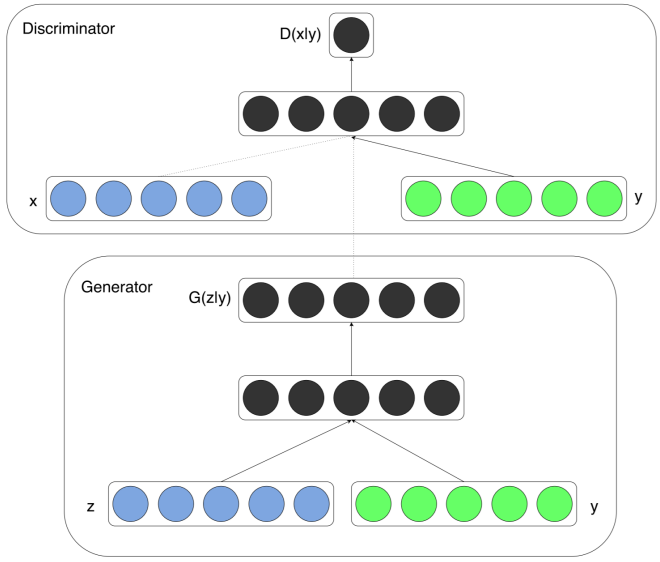


Figure 3.14: Structure overview of cGAN [46]

As the generator is conditioned on $\boldsymbol{y}$, its output depends on both the random input z and $\boldsymbol{y}$, thus $G(z \mid \boldsymbol{y})$ or often denoted as $G(z, \boldsymbol{y})$. The same holds for the discriminator, it is also conditioned on $\boldsymbol{y}$, thus $D(\boldsymbol{x} \mid \boldsymbol{y})$ or equivalent $D(\boldsymbol{x}, \boldsymbol{y})$. In terms of training, the only difference with respect to GANs is the implementation of the conditional probabilities. Thus, the training procedure shown in section 3.4.3 is followed, with the value function used in the zero-sum games as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\log D(\boldsymbol{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y},z}[\log(1 - D(G(z,\boldsymbol{y}),\boldsymbol{y})))] \tag{3.17}$$

3.6 Image-to-Image Translation

In this section, the Pix2Pix network created by Isola et al. [21] in the paper *Image-to-Image Translation with Conditional Adversarial Networks* will be explained.

Image-to-Image is an application of cGANS where the condition is in the form of an image. The architecture of the generator and discriminator was carefully chosen through testing and comparing performance for each modification of the network. Thus, the Pix2Pix network provides an general framework for various image-to-image translations, e.g., gray-to-color, day-to-night, aerial-to-map.

3.6.1 Pix2Pix Generator - U-Net

The generator utilizes a convolutional neural network architecture called U-net, established by Ronneberger et al. [47]. A representation of a U-Net can be seen in Figure 3.15. The input layer is an image; this is down-sampled on the encoder side, using convolutional layers with drop-out and batch normalization, until a bottleneck layer. At the bottleneck, the process is upsampled on the decoder side, by use deconvolutional layers. An essential aspect of this structure is the use of skip connections between the layers in the downsample and upsample side, showed by the dotted line in the figure. The skip connection is used to carry over the low-level information shared between the input and output layer. Additionally, there is no random noise vector, z; instead, some sort of randomness is introduced through the drop-out layers.

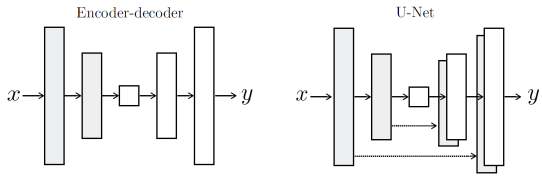


Figure 3.15: Schematics of a generator following a encoder-decoder structure [21]. The leftmost figure shows a regular encoder-decoder generator. The rightmost figure shows a U-Net generator, with skip connections between the encoder and decoder side.

3.6.2 Pix2Pix Discriminator - PatchGAN

The discriminator utilizes a concept called a PatchGAN discriminator, visualized in Figure 3.16. This is a deep convolutional neural network that scans patches of the input image rather than the entire image, as done in regular GANs. The patches are generally of size $N \times N$. However, the authors found that a 70×70 patch yielded the best result [21]. Consequently, the discriminator's output is a 30×30 matrix, where each element represents the probability of each patch being real. Therefore, the discriminator is more locally dependent, and it assumes that pixels with a diameter larger than a patch size are independent of each other. Finally, the result matrix is averaged out to provide the prediction of D. This technique enables the discriminator to effectively run on larger images with fewer parameters, thus being easier to train.

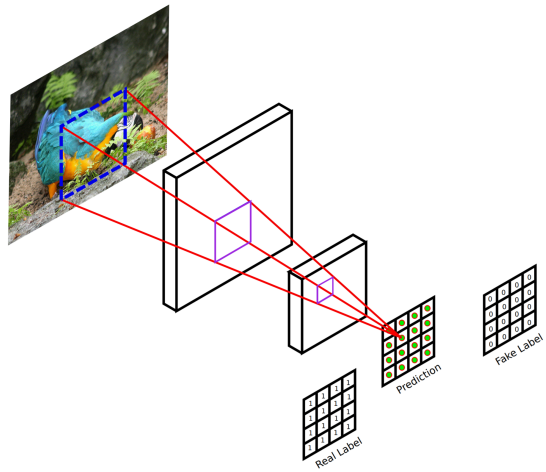


Figure 3.16: Representation of PatchGAN discriminator [48]. The discriminator evaluates patches from the image of being real or fake trough covolutional layers. The result from the discriminator is shows as a matrix containing the prediction.

3.6.3 Loss Functions and Training Process

For image-to-image translation, an optimized network can, with good accuracy, generate samples that are similar to that of the training data. However, using the value function, for cGANs given in Equation 3.17 may not sufficiently train the generator network. This value function only labels the images as real or fake. An additional loss that penalizes the generator if the pixel values are far from the real image is wanted, and thus forcing the generator to produce a real image and an image as close to the input as possible. Using either the L1-distance or the L2-distance as an additional loss will provide such penalization. However, L2-distance is prone to yield blurry images, which in the context of generating a real-looking image is not suitable. Therefore, is the L1-distance as seen in Equation 3.18 applied. Note that the latent variable, z, is present in the equation. However, this only represents the randomness created by the use of drop-out layers.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y},z}\left[\|\boldsymbol{y} - G(\boldsymbol{x}, z)\|_1\right] \tag{3.18}$$

The Pix2Pix network is optimized following the training algorithm presented in **Algorithm 1** with one step on D, and one step on G. The stochastic gradient descent algorithm *Adam* is used. Furthermore, to avoid saturation, the generator maximizes $\log(D(G(z, \boldsymbol{y}), \boldsymbol{y})$, let this be denoted as $\mathcal{L}_{cGAN}$. Combining the L1-distance with $\mathcal{L}_{cGAN}$, the generator loss is as follows:

$$\mathcal{L}_G(D, G) = \mathbb{E}_{\boldsymbol{y},z}[\log D(G(z, \boldsymbol{y}), \boldsymbol{y}))] + \lambda\mathcal{L}_{L1}(G). \tag{3.19}$$

in which λ is a hyperparameter the specifies the importance of the L1-distance to the generator loss. A $\lambda = 100$ showed to be the most effective [21]. The

discriminator loss follows the traditional cGAN loss, as seen below.

$$\mathcal{L}_{cGAN} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\log D(\boldsymbol{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y},z}[\log(1 - D(G(z,\boldsymbol{y}),\boldsymbol{y})))] \tag{3.20}$$

Chapter 4

Data Collection Method

In order to train and test the performance of the neural network used in this thesis, a data set has to be obtained. The data set consists of images of the surface elevation of pits and their corresponding stress field at a critical distance below the surface. This chapter presents the methods used for creating this data set and choices made during modeling.

4.1 Overview

The neural network is aimed to work as an effective and fast tool in fatigue assessment of mooring chains by predicting the stress field at a critical distance. As discussed in chapter 2, the fatigue life of notched specimens is reduced as the notches act as stress raisers, and thus reducing the fatigue resistance. Therefore, looking at pitting corrosion as a geometry effect, rather than a chemical process, the effects of pitting corrosion can be assessed using the theory of critical distance. In this thesis, the critical distance was assumed to be constant at a distance, $L = 1$ mm, below the surface. Following the point method, discussed in section 2.4, the stresses at $L/2 = 0.5$ mm was used in the assessment.

The entire method of creating a data set is summarized in the flowchart in Figure 4.1. This data set contained two types of images; one describing the surface elevation of a pit, and one describing the stress field at the critical distance. The latter was used to evaluate the network performance of predicting the stress field, whereas the first was used as the condition for the network. The surface elevation images were based on 3D scans of real pits on corroded mooring chains. By use of a finite element model a method of creating the stress field images was established. In the following, a description of how the geometry of the pit was obtained, how the finite element model was established, and lastly, how the data set was created is given.

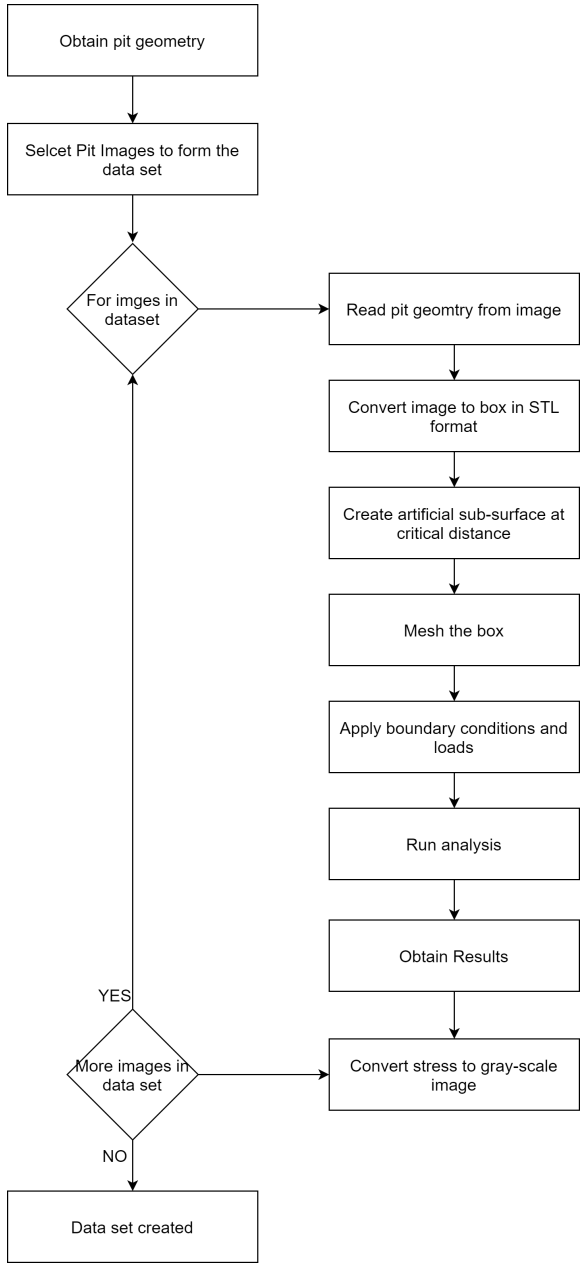


Figure 4.1: Workflow for data collection method

4.2 Obtaining Pit Geometry

The geometry of individual pits was provided by Marius Andersen. The exact method for 3D scanning mooring chains and the method used in the post-processing of these scans was not provided. However, a short description of the procedure for creating images containing the pits' surface elevation will be given.

In order to analyze the effect of one pit, the post-processed 3D scans were used to sort out single pits with their corresponding geometry and location on the chain. By using the depth and size of the pits, a pit that fitted within a small patch could be detected. Furthermore, when transforming the pit surface to an image, the extracted pit was placed in a 60×60 mm^2 patch, where the surrounding surface of the pit was assumed flat, not curved as the actual surface of the chain. Moreover, the pit was oriented in such a way that the expected maximum stress was to occur in the x-direction in the surface image. Figure 4.2 shows an example of a pit extracted from a chain and transformed into an image is showed. The red patch is oriented such that the expected maximum stress is to occur in the x-direction. The horizontal direction in the image corresponds to this x-direction. The pit surface image was a 8-bit gray-scale image with 240×240 pixels. A scale factor based on a maximum depth of 11.93 mm was used to represent pit depths consistently. The maximum pixel value of 255 corresponded to a depth of 11.93 mm.

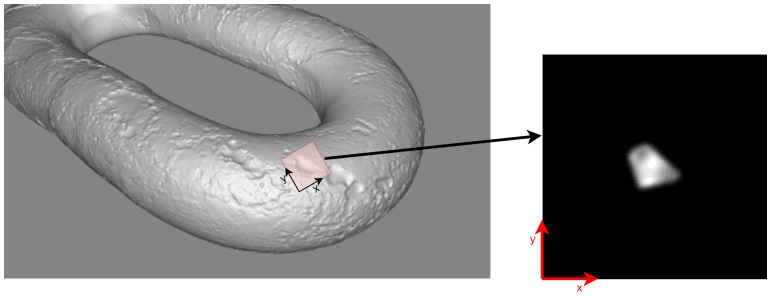


Figure 4.2: Example of pit being extracted from 3D-scan. Note: the image on the right is scaled in order to visually be able to see the pit.

However, only one continuous pit was transformed into a surface image. This means, if two different pits with no connection were to fit within the small patch, these two would be separated into two different images, see Figure 4.3. In this way, a robust method of investigating the effects of the pits could be established.

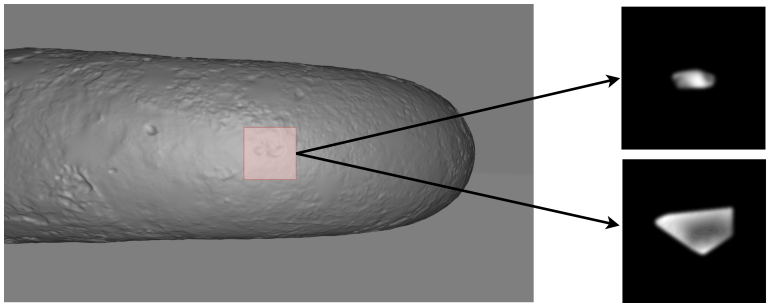


Figure 4.3: Example of two pits within one patch during extraction of single pits. Note: the images on the right is scaled in order to visually be able to see the pit.

4.3 Finite Element Model Procedure

A finite element model was used to obtain the stress field at the selected critical distance. The model was based on the pit surface images. Thus, the dimensions used to create the image had to be carried over to the FE model. Moreover, as the neural networks needs large data sets, i.e., a large number of images, to be trained sufficiently, running the finite element method in batch mode was established. This method, referred to as the finite element model procedure, is visualized in Figure 4.4.

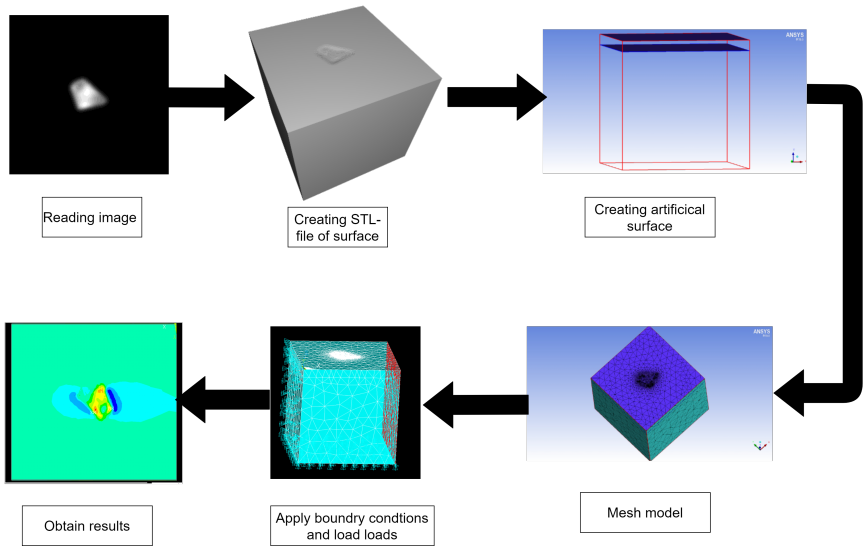
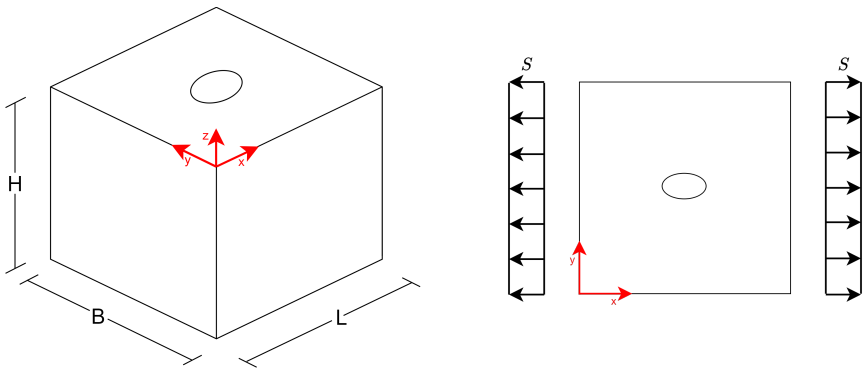


Figure 4.4: Overview of the finite element model procedure

The finite element model consisted of a $60 \times 60 \times 60$ mm^3 box, as seen in Figure 4.5a. The top surface of the box was used to replicate the pitted surface. The box model was created by essentially imposing the pit image on the top surface and then use the surface elevation as a negative height, i.e., the pit goes into the box. The process of creating a STL-file of the box model is seen as the two first steps in the Figure 4.4. Using a $60 \times 60 \times 60$ box, enabled the model not to let the stress in close proximity to the pit be affected by the applied boundary conditions at the bottom and sides, as these could cause a local stress increase.



(a) 3D view of the model **(b)** 2D - Top view of box

Figure 4.5: Representation of box model used in the finite element model procedure

Table 4.1: Material constants and dimensions in FE model

	Value	Unit	Description
E	209	GPa	Youngs modulus
ν	0.3		Poissons ratio
H	60	mm	Box height
L	60	mm	Box length
B	60	mm	Box height
S	1000	Pa	Applied stress

To obtain the stress field at the critical distance, an artificial surface was created, seen as the third step in Figure 4.4. The artificial surface was identical to the pit surface. This enabled the procedure to extract the stress precisely at a distance of 0.5 mm below any point along the pitted surface. As know from chapter 2, the stresses in close vicinity to geometry defects decrease rapidly. Thus, to describe these rapid changes, a fine mesh was needed. Using *ANSYS ICEM CFD 19.0*, hereinafter referred to as ICEM, as a meshing tool, a meshing procedure that used a proximity-based refinement of the mesh was utilized. These refinements empowered a fine mesh in areas with complex geometry, such as the pit region, and a coarser mesh where the geometry was more modest. The model was meshed

with 4-node tetrahedral elements, named SOLID 185 in ANSYS, in which each node has three degrees of freedom; translation in all three directions, x,y and z.

Furthermore, the meshed box was analyzed by use of *ANSYS Mechanical APDL R.19*, referred to as ANSYS. As specified in the provided image, the assumed maximum stress was in the x-direction. Hence, the box was placed under uniaxial stress load going in x-direction, as illustrated in Figure 4.5b. The applied material properties and box parameters are shown in Table 4.1.

The procedure for creating the FE model consisted of using MATLAB to run the different tasks in batch mode. MATLAB was used to create an STL-file of the box model based on the pit surface image, and ICEM was used to create the artificial sub-surface and mesh the model. Finally, ANSYS was used to apply loads and run finite element analysis.

4.4 Creating the Data Set

The neural network was intended to predict the result stress field image at a critical distance. Therefore, stress field images were created to be used as training samples for the network. These stress field images were created by mapping the nodal stresses on the artificial sub-surface into a 240×240, 16-bit gray-scale image. Each pixel represented the corresponding critical stress at the same location as the image of the surface elevation.

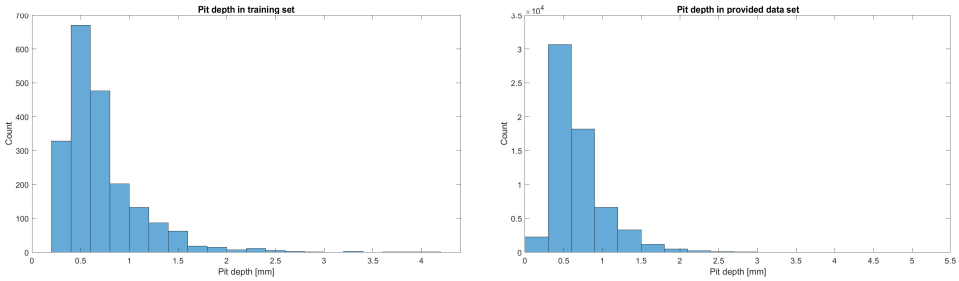
However, as the applied load on the box had no meaning other than resulting in stress concentration around the pit, the stress concentration factor at the critical distance was chosen as the value represented by the pixel values. The stress concentration at the critical distance is, for the rest of this thesis, referred to as the stress concentration factor or SCF.

Furthermore, to utilize the generalization ability of neural networks, the training data has to be a good representation of the whole problem domain. To create a robust network, the data set should include unexpected or unrealistic data points. Relating this to a SCF; a factor of 5 yields the very upper end of the expected stress concentration factors, which represents sharp notches acting more as cracks rather than notches. To include the possibility of correctly generate samples with unexpected values, the maximum stress concentration factor in the stress field images was limited to 10. Which in terms, enables the network to be correct down the fourth decimal in the SCF value. The transformation from local stress value σ, to the pixel value P, or finding the SCF based on the pixel value, was performed using Equation 4.1, in which S was the applied stress presented in Table 4.1.

$$P = \frac{\sigma}{S} \cdot \frac{2^{16} - 1}{10} = \text{SCF} \cdot \frac{2^{16} - 1}{10} \tag{4.1}$$

As mentioned above, the data set has to be a good representation of the whole problem domain. In this present thesis, approximately 65000 different images of

pits were provided to the author. Using all these images in the network's training is desired; however, such a large data set would be extremely time-consuming to train with. In addition, running all the images through the finite element model procedure would further increase the total time consumption from start to finish. Hence, 2028 randomly selected pits were used to create the training data set, and 418 images were used to create the test set. As discussed in chapter 2 the stress notch effect factor is dependent on the pit depth. A comparison of the pit depth distribution in both the training and test set was conducted. In Figure 4.6 a histogram plot of the pit depths in the entire data set and selected data set is provided.



(a) The selected training data set **(b)** The whole provided set

Figure 4.6: Histogram of the pit depth distribution in the selected training set, and the entire provided data set

The finalized input for the neural network can be seen in the figure below. During the training of the neural network, the network will use the input image alongside the stress field image to calculate the losses and thereby optimize the generator and the discriminator. To ensure that the correct input image was paired with the resultant stress field, the two images were concatenated, as seen in Figure 4.7.

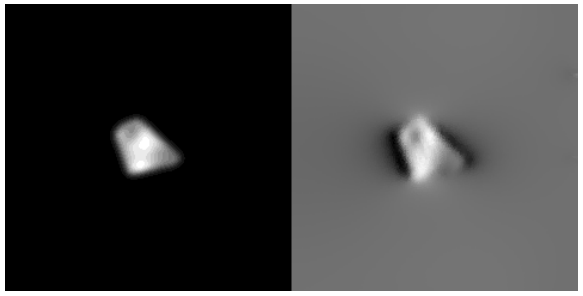


Figure 4.7: Input image for the neural network. The left image is the pit surface elevation, the right image is the stress field at the critical distance. Note: Both images has been locally scale to be able to visually see the them.

Chapter 5

Implementation of Pix2Pix Network

In this section, the implementation of the machine learning algorithm, Pix2Pix, discussed in section 3.6, is presented. It should be emphasized that the created network, and its architecture, was based on the original Pix2Pix network[21] and the TensorFlow tutorial on Pix2Pix[1]. Applying an algorithm created by others, requires some customization and trial and error to fit the desired problem. Furthermore, to distinguish the neural network used in this thesis[2] from the original network, the latter is referred to as Pix2Pix, whereas the first is named *Pit2Pix*.

5.1 Environment Setup

The neural network was chosen to use the open-source library within machine learning named TensorFlow. The library was created by the Google Brain Team and made public in November 2015. TensorFlow supports programming languages such as Python, C++, JaveScript, and more as the front-end, making TensorFlow a versatile tool applicable for a vast number of tasks. All mathematical operations are channeled to be performed in the high-performance C++. In addition, TensorFlow is highly compatible with other libraries such as Keras. Hence, enabling a high-level functionality and making it simpler to create neural networks and deep learning models without the need for hardcoding every single mathematical operation, such as those presented in section 3.2.

Moreover, the machine learning code was created using the cloud-based Google Colaboratory, for short Colab. Colab allows the user execute Python programming language directly in their browser. The Python code was run in virtual machines

[1] `https://www.tensorflow.org/tutorials/generative/pix2pix`

[2] `https://drive.google.com/drive/folders/1tEUJTSvg6KwNQPOlrlzaP5Bp5T0gHF3K?usp=sharing`

operated by Google and used Jupyter Notebook as interface. Using Colab alongside TensorFlow was immensely beneficial, as one can use graphics processing units (GPUs) or tensor processing units (TPUs). This massively increased the virtual machines' computational power, and thus decreased the time consumption of training the neural network. In addition, established codes, networks, and data set could easily be shared and run across computers by saving the data in Google drive.

5.2 Building the Neural Network

Building the neural network consisted of creating an input pipeline, to efficiently retrieve the training data. Then, further creating the generator and discriminator, and lastly specifying the loss functions, optimization algorithms. These steps will be explained below.

5.2.1 Input Pipeline

The data set used for training and testing the network consisted of images as that seen in Figure 4.7. These concatenated images ensured that the correct images were paired. However, the input image was separated into two, as only the leftmost part, pit surface elevation, was used as the condition to both the generator and discriminator. The rightmost part, the stress field image, was used in the training process, as these images represented the distribution space of the stress field images. In addition, the stress field image was used to calculate the L1-distance.

The network architecture was designed to process images of size 256×256. To comply with this, the two separated images were resized from the original 240×240 size to 256×256. In addition, these up-scaled images were normalized between -1 and 1, which is a common practice in machine learning.

5.2.2 The Generator

The generator architecture followed the U-Net architecture described in section 3.6.1. The down-sampling side, known as the encoder, used convolutional batch- normalization layers with LeakyReLU activation functions, with a slope $a = 0.2$. Each convolutional layer used k filters with size 4×4 and stride $s = 2$, which resulted in a down-sampled factor of 2 for each layer. Let one layer of these layers described above be denote as Ck. The encoder architecture was as follows: C64-C128-C256-C512-C512-C512-G512. As an exception, the first layer did not use batch-normalization.

At the up-sampling side, the U-net decoder, deconvolutional batch-normalization layers with a dropout rate of 50 % was used. ReLU activation function was used in the three first layers. Each of these layers had k filter with a size of 4×4 and strides $s = 2$, resulting in a up-sampling factor of 2. Let one such layer denote as CDk. The last four layers used a convolutional batch-normalization with ReLU activation function, and filter as described above. Let this be denoted as, CRk. The U-net

decoder architecture was as follows: CD512-CD512-CD512-CR512-CR256-CR128-CR64.

The U-Net utilized skip connections between a layer i in the encoder and a layer $n - i$ in the decoder, where n denotes the total number of layers. The output of one layer i was concatenated with layer $n - i$. Thus, doubling the input dimension on the layers in the U-net decoder. At the last layer, a convolution was applied to match the size of the output image and channels, with a following Tanh activation function. Figure 5.1 gives an overview of the generator architecture. The yellow color indicates a result of a convolutional batch-normalization layer with LeakyReLU, whereas the blue color indicates the result from a deconvolutional batch-normalization dropout layer with ReLU. The blue lines with arrows indicates the skip connections.

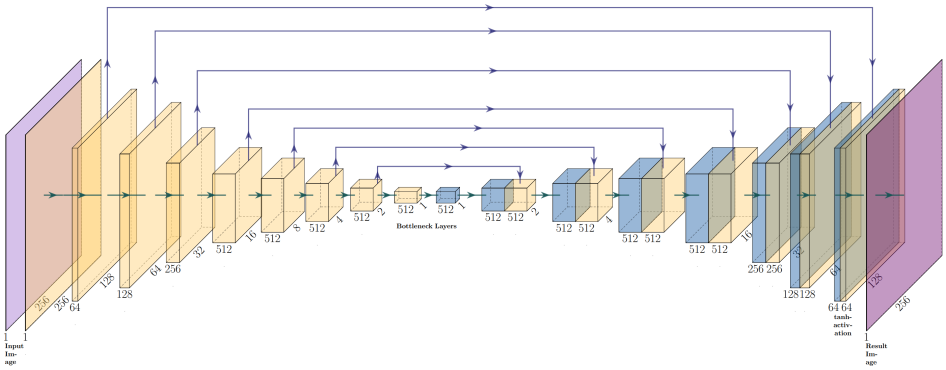


Figure 5.1: Visual representation of implemented generator architecture. The yellow color indicates a result of a convolutional batch-normalization layer with LeakyReLU, whereas the blue color indicates the result from a deconvolutional batch-normalization dropout layer with ReLU. Blue lines with arrows indicates the skip connections.

5.2.3 The Discriminator

The discriminator was a patchGAN, as discussed in section 3.6.2. The size of the patches was 70×70 pixels resulting in a 30×30 output matrix. Two images were used as input to the discriminator, namely the generated image and the real target image. All layers in the discriminator were convolutional batch-normalization layers with leakyReLU activation function with a slope of 0.2. The layers had k filters with a size of 4×4 and stride $s = 2$, again denoted as Ck. Between the last two, a zero-padding was employed. At the last layer, a convolution was made to output the 70×70 matrix. The discriminator architecture was as follows: C64-C128-C256-C512. Figure 5.2 shows an overview of the patchGAN where the yellow box is the result of a convolutional layer, and the red box indicates a zero-padding layer.

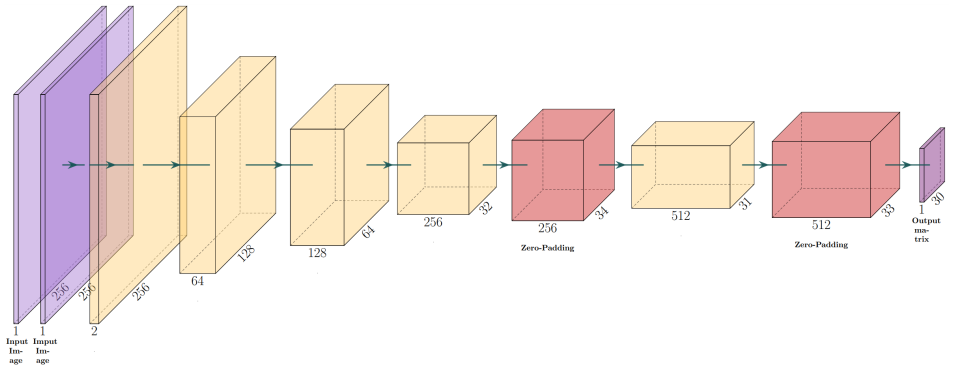
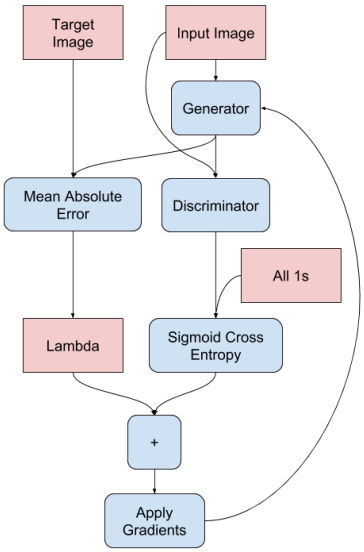


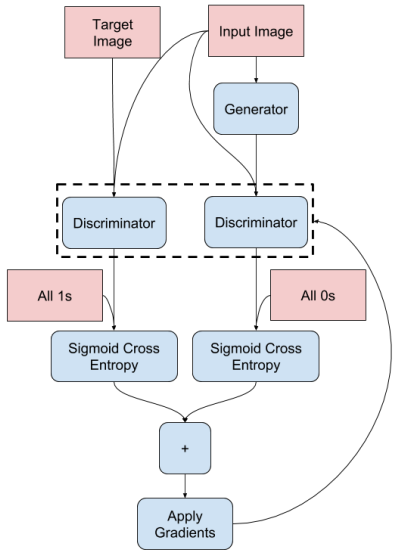
Figure 5.2: Visual representation of the implemented discriminator architecture used in the neural network. The input layer, the purple color, is a generated image and a true image. The yellow box is the result of a convolutional layer, and the red box indicates a zero-padding layer.

5.2.4 Loss Functions and Optimization

The applied loss function is as specified in section 3.6.3. The generator loss and the discriminator loss was as shown in Equation 3.19 and Equation 3.20, respectively. The optimization procedure followed the algorithm presented in **Algorithm 1**, with one step on D, and one step on G. Both updates used the gradient descent algorithm, *Adam*, with momentum parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$ and a learning rate $\eta = 0.0002$. Figure 5.3 shows an visual representation of the optimization process as performed in the neural network code is presented. As the correct label on the input images feed the discriminator was known, the cross entropy loss was calculated between the output matrix from the discriminator and a matrix of same dimension, i.e., 30×30, contain either ones or zeroes depending on the true or false label respectively.



(a) Generator training process. The D loss was calculated as $\mathbb{E}_{\boldsymbol{y},z}[\log D(G(z,\boldsymbol{y}),\boldsymbol{y}))]$, and the mean absolute error was the L1-distance loss. The SDG algorithm *Adam* was used to calculate the gradients.



(b) Discriminator training proces. The D loss is calculated as presented in Equation 3.20.The SDG algorithm *Adam* is used to calcualte the gradients.

Figure 5.3: Overview of training process of the implemented network [49]

5.3 Maximum Loss and Other Modifications

As discussed in chapter 2, the highest stress concentration has most weakening effect. Also, using the position, the highest SCF can yield a potential failure point. Thus, to further improve the generator's ability to accurately predict the highest SCF, an additional loss for the generator was used. This loss can be looked upon as a maximum value loss, where the generator was penalized when the maximum value of the generated image, $\max(G(x,z)_{i,j})$, differs from the maximum value in true stress field image, $\max(y_{i,j})$, see Equation 5.1. In this case, the total generator loss was as seen in Equation 5.2, the maximum loss is scaled with $\lambda/2$.

$$\mathcal{L}_{max} = \mathbb{E}_{x,y,z}\left[\max(y_{i,j}) - \max(G(x,z)_{i,j})\right] \tag{5.1}$$

$$\mathcal{L}_G = \mathcal{L}_{cGAN} + \lambda\mathcal{L}_{L1} + \frac{\lambda}{2}\mathcal{L}_{max} \tag{5.2}$$

Chapter 6

Results

In this chapter, the results from the neural networks called *Pit2Pix* and *Pit2PixV1* are presented. The neural networks were trained on the training set in the ImgResComb.zip folder[1]. In order to compare the results from the two different network configurations, training losses and the generated stress field images are presented. In addition, image metrics as mean absolute error and standard deviation between the generated samples and the true samples, alongside the maximum SCF prediction error, are shown.

6.1 Time Spent During FEA

The time consumption of creating one stress field image, which followed the finite element model procedure presented in section 4.3, is presented in Table 6.1. This can be separated into two, the time spent in ICEM for creating artificial subsurface and mesh the box, and the time spent running FEA in ANSYS. The time spent creating both training and test data set, containing 2028 and 418 samples respectively, is presented in Table 6.2.

Table 6.1: Time consumption of creating one stress field image

Data set	ICEM time [min]	ANSYS time [min]	Total time [min]
Training set	3.5	3.5	7
Test set	3.5	3.5	7

Table 6.2: Total time consumption of data set creation

Data set	Images	Total time [days]
Training set	2028	10
Test set	418	2

[1] `https://folk.ntnu.no/haakonkp/PitsImgComb.zip`

6.2 Pit2Pix Performance

This section presents the results obtained from training and testing the neural network named *Pit2Pix*. The network was build as specified in section 5.2. The training losses, image metrics, and examples of generated images are presented. Training the network was performed over several different sessions. Many of these training sessions did not yield satisfactory results, thus results from these will not be presented in this present thesis. However, losses from all training sessions can be found on TensorBoard[2].

The best-performing model was evaluated on two criteria. First was the training losses. The different training losses should be stable over the entire training session, and they should converge to. In addition, low L1 distance loss was wanted. The second criterion was the quality of the images, i.e., producing real-looking images, and especially for this thesis, accurately estimate the highest pixel value in the image, hence the highest SCF in the image.

6.2.1 Image Metrics

The result from testing *Pit2Pix* on the 418 samples in the test set is presented below. The actual maximum SCF in each of the true stress image versus the predicted maximum SCF in each of the generated image is presented in Figure 6.1. The transformation from pixel value to SCF followed Equation 4.1. The red line in the plot shows $x = y$, meaning that both the prediction and the actual value are the same.

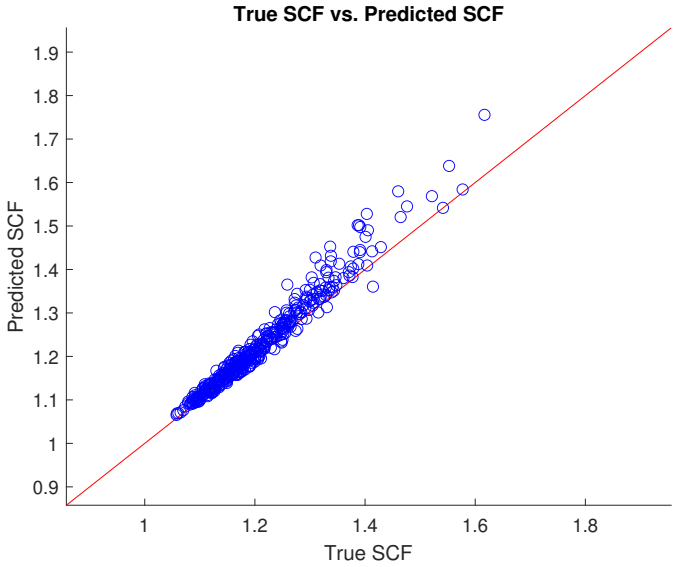
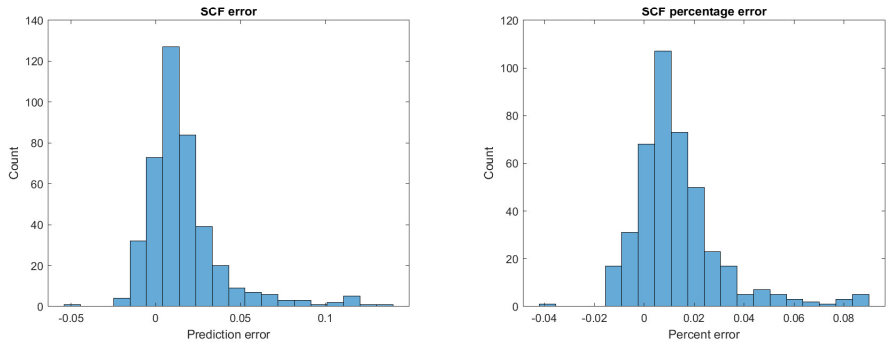


Figure 6.1: True maximum SCF vs. redicted maximum SCF of *Pix2Pix*. The red line shows $x = y$, i.e., the predicted and true maximum SCF is identical.

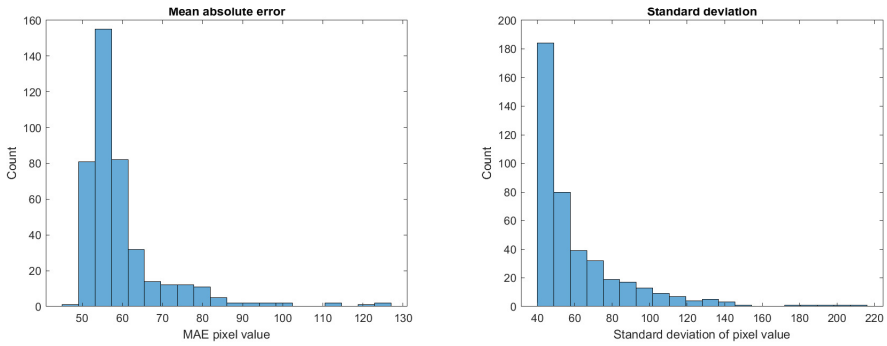
[2] `https://tensorboard.dev/experiment/sGlXOOAIRjOQlvzmUX6C1Q/`

The prediction error in maximum SCF, and the percentage error, of each of the 418 samples are presented in Figure 6.2. The average deviation was 0.0169. In addition, considering the whole image, the mean absolute error in pixel value between the generated stress field images and the real stress field images, with its standard deviation, is presented in Figure 6.3.



(a) SCF error **(b)** SCF error in percent

Figure 6.2: *Pit2Pix* network histogram of maximum SCF error



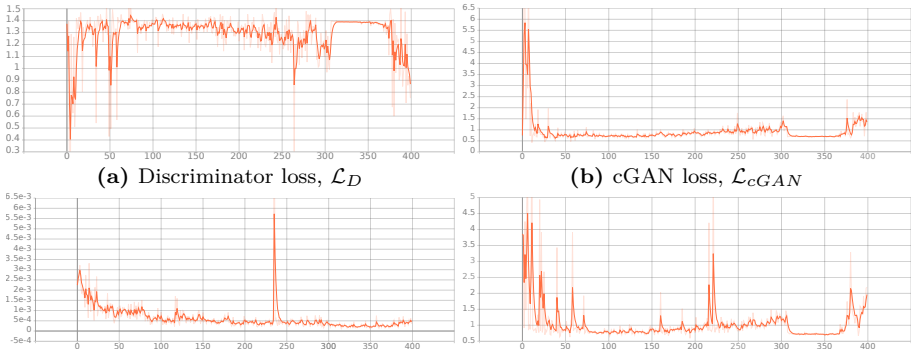
(a) Mean absolute error **(b)** Standard deviation error

Figure 6.3: *Pit2Pix* Network image performance metrics

6.2.2 Losses

The discriminator losses, generator cGAN losses, L1-distance losses, and total generator losses obtained during training, are presented in Figure 6.4. The actual values of the discriminator and generator losses are not that important, as only the optimized point in the adversarial training is wanted. Only values in the L1-distance losses are of direct interest, which ended with a value of $4.6 \cdot 10^{-4}$. However, note that the images were normalized between -1 and 1 in the input pipeline. The network was trained over 400 epochs, with a batch size of 1. One epoch is a com-

plete run through the entire training set. The training time for one epoch varied between 65 seconds to 120 seconds depending on the available GPUs in Colab. The particular time for the training session, with 400 epochs, for this network was 8 hours.



(a) Discriminator loss, $\mathcal{L}_D$ **(b)** cGAN loss, $\mathcal{L}_{cGAN}$

(c) L1 loss, $\mathcal{L}_{L1}$ **(d)** Total generator loss, $\mathcal{L}_G$

Figure 6.4: Training losses for *Pit2Pix*. The horizontal axis show the epochs, whereas the vertical axis is the loss value.

6.2.3 Example Images

To evaluate the generator's ability to generate images that are close to the ground truth, a visual inspection of the images is needed. Additionally, the reader gets an insight into the generator's prediction power. In Figure 6.5 randomly selected examples of the generated stress-field image alongside their input image, true stress field image, and the difference between the generated and true stress image is shown. The difference image is the absolute difference between each pixel in the image. Plase the GoogleDrive for more examples[3]

[3] `https://drive.google.com/drive/folders/1jmo48dDOWjmaJ0x_Y_S74t_YYjktPuE-?usp=sharing`

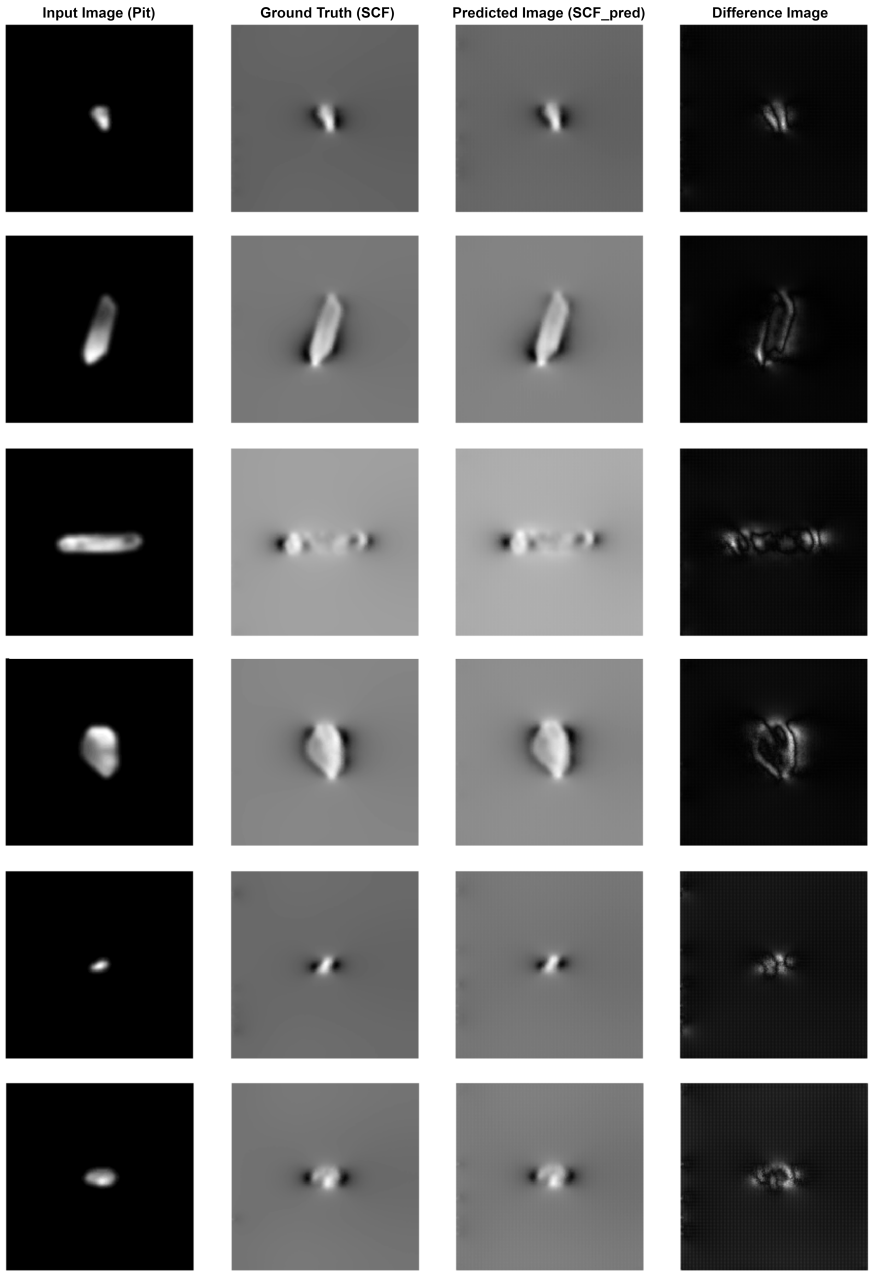


Figure 6.5: Examples of generated images of the *Pit2Pix* network. The image on the left is the input image to the generator. The middle-left image is the true stress field image, whereas the middle-right image is the generated image. The rightmost image is the difference image between the true and predicted image. Note: All images are scaled to be visually able to see them.

6.3 Pit2PixV1 Performance

In this section, the result from the network *Pit2PixV1* during training and testing is presented. This network configuration utilized the additional max loss for the generator, as described in section 5.3. In addition, the network was tested on the same 418 samples in the test set, to be able to compare the two networks.

6.3.1 Image Metrics

The predicted maximum SCF and the true maximum SCF of the 418 samples in the test are presented in Figure 6.1. The transformation from pixel value to SCF followed Equation 4.1. The red line shows $x = y$, i.e., the predicted maximum equals the true maximum.

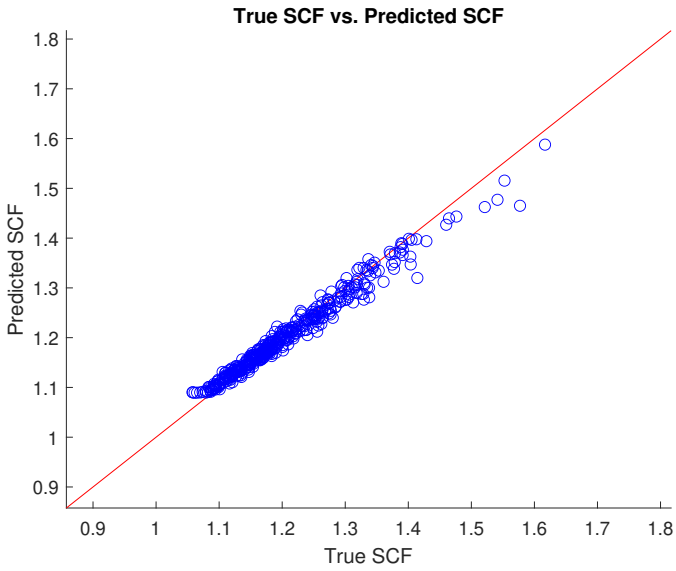
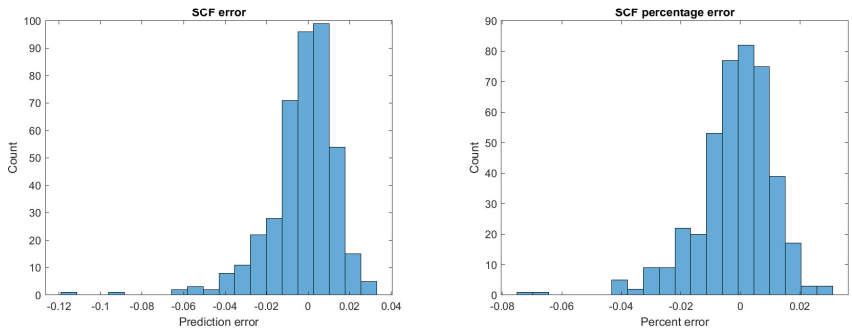


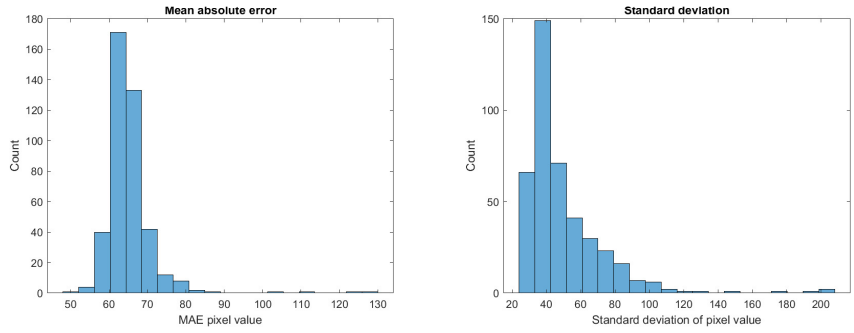
Figure 6.6: True maximum SCF plotted against predicted maximum SCF of *Pix2Pix*. The red line shows $x = y$, i.e., the predicted and true maximum SCF is identical.

In Figure 6.7 the maximum SCF error and the percentage error in SCF is shown. The mean deviation was -0.0028 To evaluate the difference between the true and predicted image as a whole, the mean absolute error in pixel values and the standard deviation is shown in Figure 6.8



(a) SCF error **(b)** SCF error in percent

Figure 6.7: *Pit2PixV1* network histogram of maximum SCF error

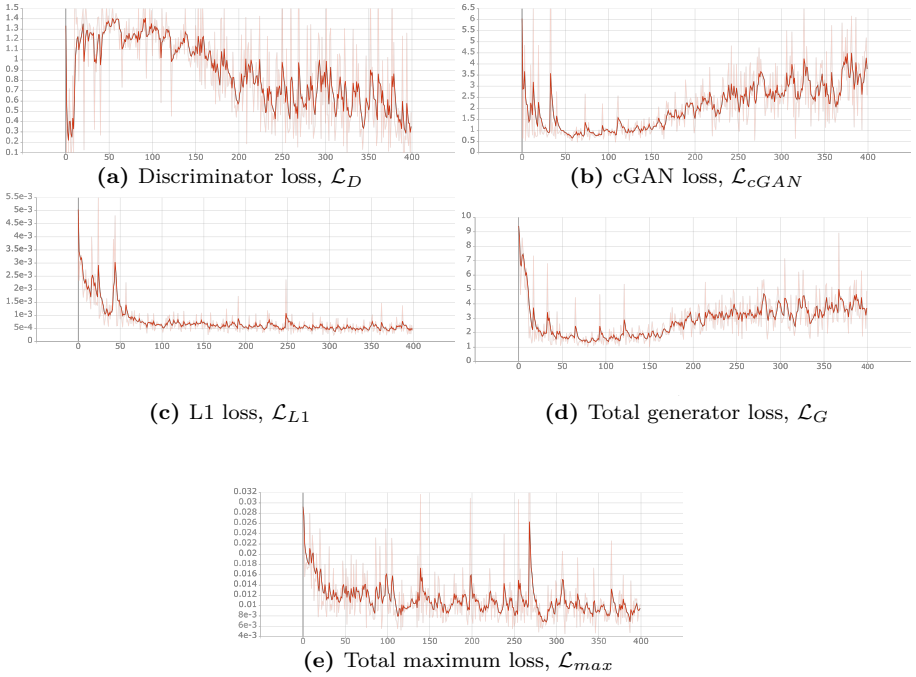


(a) Mean absolute error **(b)** Standard deviation error

Figure 6.8: *Pit2PixV1* network with max-loss image performance metrics

6.3.2 Losses

The exact loss values are not that important, as only the optimized point is of interest for the generator and the discriminator. However, as mentioned above, the values L1-distance loss is of interest, where a low value is wanted. Moreover, the maximum loss should be small, resulting in a more accurate maximum SCF value. In Figure 6.9 the different losses obtained during training are shown. In this training session, the total time to finish training over 400 epochs was 11 hours, and the L1-distance loss and max loss ended up with a value of $5.18 \cdot 10^{-4}$ and $9.95 \cdot 10^{-3}$ respectively.



(a) Discriminator loss, $\mathcal{L}_D$

(b) cGAN loss, $\mathcal{L}_{cGAN}$

(c) L1 loss, $\mathcal{L}_{L1}$

(d) Total generator loss, $\mathcal{L}_G$

(e) Total maximum loss, $\mathcal{L}_{max}$

Figure 6.9: Training losses for *Pit2PixV1*.The horizontal axis show the epochs, whereas the vertical axis is the loss value.

6.3.3 Example Images Generated

To evaluate the *Pit2PixV1* generator's ability to generate real-looking images that are close to the true stress field image, a visual inspection of the generated images are useful. The visual inspection was additionally used to see if the max loss affected the prediction accuracy on the whole image. Plase the GoogleDrive for more examples[4]

[4] `https://drive.google.com/drive/folders/18CrKq784ZNHf92fN2sj6MoOYfCbTA8l3?usp=sharing`

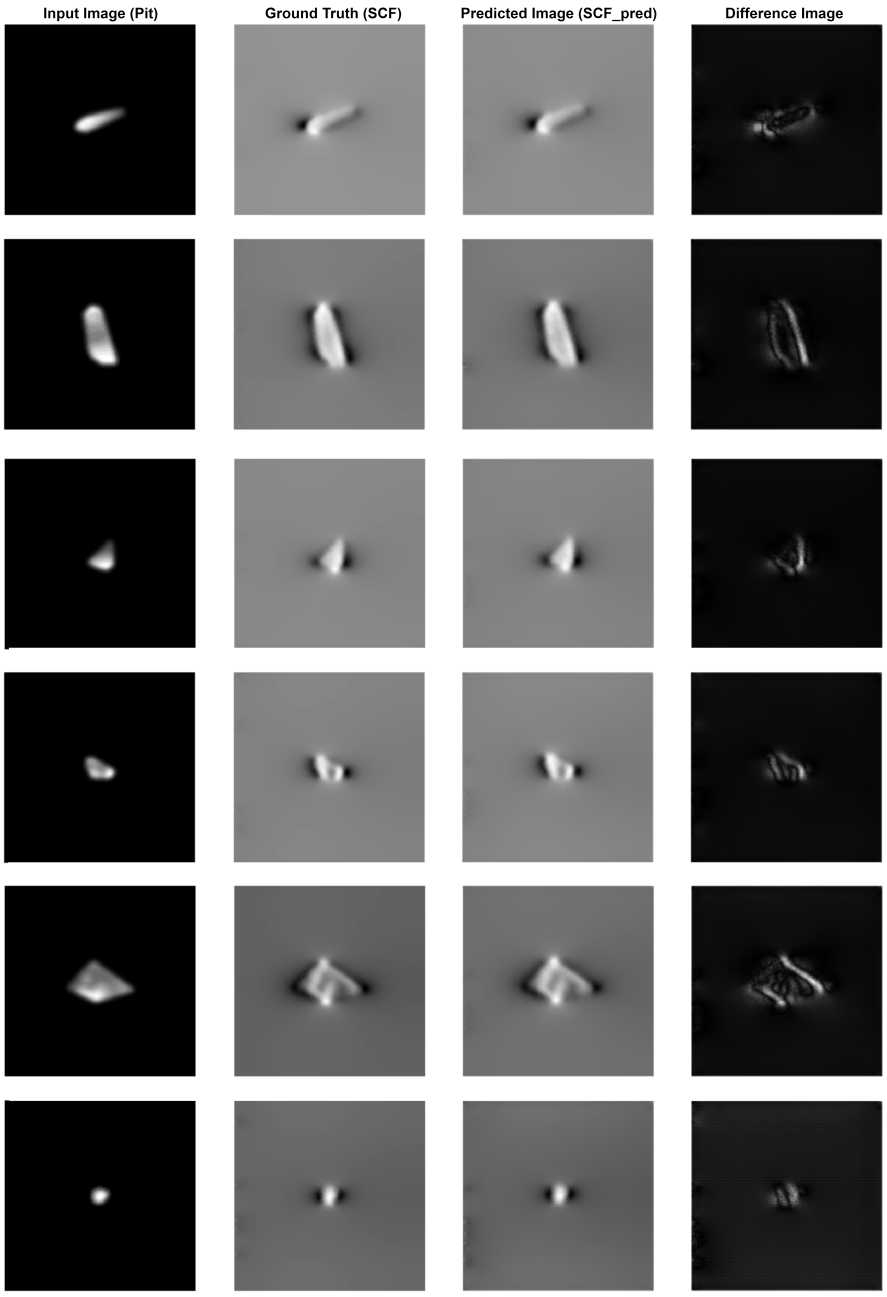
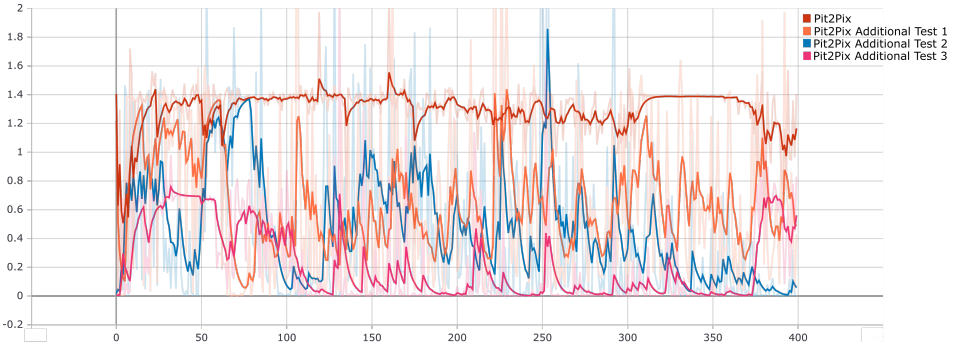


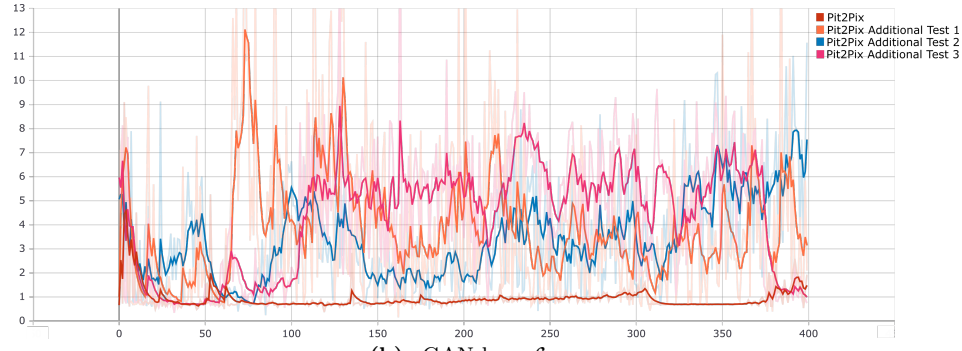
Figure 6.10: Examples of generated images of the *Pix2PixV1* network. The image on the left is the input image to the generator. The middle-left image is the true stress field image, whereas the middle-right image is the generated image. The rightmost image is the difference image between the true and predicted image. Note: All images are scaled to be visually able to see them.

6.4 Training Instability

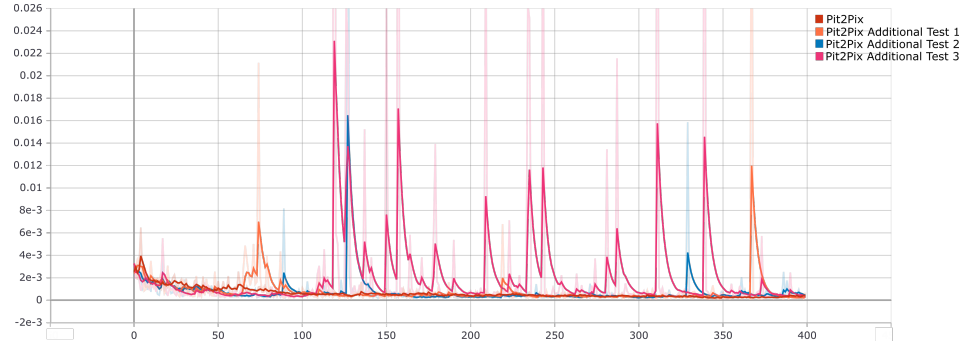
The *Pix2Pix* network was trained and tested multiple times to find the best performing network. The best network shows a steady training process, and the generator loss and discriminator loss converges to some number. In Figure 6.11 the losses obtained during four different training sessions of 400 epochs are presented to see the difference in stability in training losses. It should be pointed out that value values have been smoothed to more easily see the trends in the training stability in each training session.



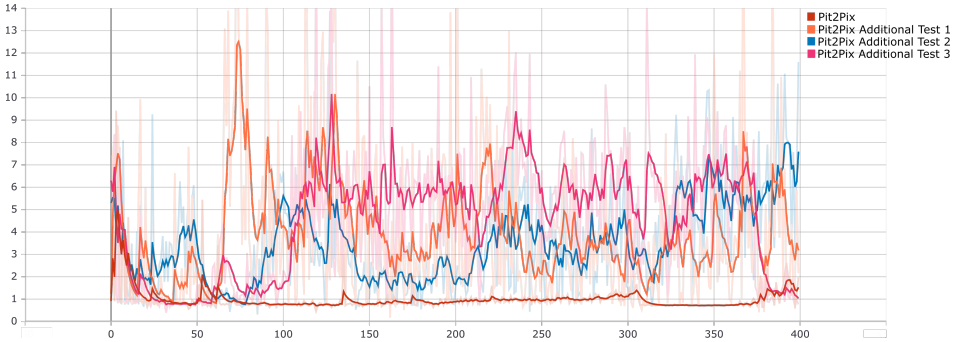
(a) Discriminator loss, $\mathcal{L}_D$



(b) cGAN loss, $\mathcal{L}_{cGAN}$



(c) L1 loss, $\mathcal{L}_{L1}$



(d) Total generator loss, $\mathcal{L}_G$

Figure 6.11: Multiple training losses for *Pit2Pix*, showcasing the instability and inconsistency with training. The horizontal axis show the epochs, whereas the vertical axis is the loss value.

Chapter 7

Discussion

7.1 Finite Element Model and Time Consumption

When modeling in ANSYS, or any other numerical solver, the choice of elements, boundary conditions, and loads affect the results. Regarding the choice of element type, the tetrahedral SOLID185 was chosen. This element type is not recommended by ANSYS, as the model consists of over 10 % tetrahedral element. This is because hexahedral elements perform better than tetrahedral as the latter can, under certain loads and boundary conditions, lock up. However, as the FE model only accounts for uniaxial tensile stress and the box model was sufficiently larger than the pit itself, it limits the possibility that the elements lock up. In addition, the boundary conditions do not affect the stress in close vicinity to the pit.

Furthermore, creating an automatic meshing routine in ICEM that meshes the surface of multiple different pits with complex geometry was more manageable with tetrahedral elements. Using other higher-order elements as SOLID186 and SOLID187, which can be hexahedral, was not an option in ICEM. Another aspect is that the higher-order elements would be more time-consuming to mesh and analyze due to an even larger number of nodes. As a data set with over 2000 images was to be created, it was not sufficient to have a time-consuming method to generate the images for the data set. Therefore, using the SOLID185 element was suitable.

The presented time consumption in Table 6.1 and Table 6.2 showed to be relatively high. Using the proximity-based refinement enabled the pit surface to be adequately meshed. The stresses in close proximity to the pit are described sufficiently, which is desired as the stresses at the critical distance are wanted. However, using the proximity-based refinement introduced a large number of elements between the top surface and the artificial sub-surface, as this was placed only 0.5 mm below the top, and thus, increasing the time spent in both ANSYS and ICEM.

7.2 Performance of Pit2Pix and Pit2PixV1.

As presented in Figure 6.1 and Figure 6.6 the two networks performance in predicting the maximum SCF are quite good. However, for larger SCFs, the prediction deviates more. As seen in the plots, most of the data samples in the test set are place within 1.1 to 1.4 in SCF. It is known that a neural network has limited capability to perform well far away from the training points. This may indicate that most training samples are placed within the region where the networks are more accurate.

Moreover, the generator's goal is to find the distribution of the input data space. It is easier for the generator to find this distribution if the data points are covering all the sample space. Also, having many samples implies that the generator has more samples to train on which provides better basis to capture the distribution the distribution. The distribution of pit depth for all the provided pit images and the selected images used in the data set can be seen in Figure 4.6. It is apparent that that majority of the pit depths in both sets are below approximately 0.8 mm. Therefore, as there are fewer training samples at the upper end regarding the pit depths, the generator will better perform within the region with many samples than outside this region. As the pit depths are one of the most weakening factors of notches, which causes larger SCFs, the distribution of pits depths indicates that the deviation at the upper end in Figure 6.1 and Figure 6.6, originates from the data set rather than the generators ability to predict these SCFs. With more training samples containing a higher SCF, the generator can have better performance.

To better cover the whole input domain, artificial pits, based on provided pits geometries, can be scaled to introduce higher SCFs. Even artificially created pits with unexpected geometry can be included in the data set. Not only does this enhance the performance at the upper end, but it also generalizes the network to all possible pits geometries. It makes the network perform well for unexpected pit shapes or other damages on the mooring chains that affects the fatigue performance.

Looking at the SCF error, Figure 6.2 and Figure 6.7 it is clear that the Pit2Pix network has an error positioned on the positive, the majority of the error are positive. This means that the network overestimates the maximum SCF in the image. On the contrary, the Pit2PixV1 network has more zero-centered error, with most of the error counts on the negative side, meaning that the network is more accurate in max SCF prediction but tends to underestimate the value.

The mean absolute error and standard deviation in pixel value of the whole image is rather small, see Figure 6.3 and Figure 6.8. As a reference, a MAE at 60 in pixel value corresponds to an error of $9.15 \cdot 10^{-3}$ in SCF. However, a low value only shows that the generated image is close to the true image as a whole. In Figure 6.5 and Figure 6.10, the difference between these two images are shown. Most of the difference in the images is found in the relatively small area under the pit. Thus, the error in the most interesting part of the image is higher than the whole image. Moreover, the example images show that the generator can generate images that are similar to the real image, which is positive.

The *Pit2PixV1* network aimed to penalize the generator if the maximum pixel value deviated. As seen in the result, the error in maximum SCF decreases, which is as wanted. However, this penalization turned the network to underestimate the stress concentration, which in contrast is undesirable. This may indicate that the penalization was too high. A lower weight on the loss can be used to reduce this. When performing a fatigue assessment, it is better to be conservative, i.e., it is better to overestimate the stresses. Hence, using *PitPix* instead of *Pit2Pixv1* may be reasonable. However, with tuning of the As the prediction error when using the maximum loss is

7.3 Training Losses and Training Instability

Training losses for the two networks was stable as presented in Figure 6.4 and Figure 6.9. The discriminator loss, and total discriminator loss shows tendency to converge to steady number. However, the numbers of the training losses are not informative. In the adversarial training procedure the generator is competing against the discriminator, this means that the losses only show relative performance of D against G. An increase in the generator loss results in a decrease in the discriminator loss.

Furthermore, the only values of direct interest are the L1 loss, and in addition, the maximum loss for *Pit2PixV1*. Both losses decreased early in training, which is positive. This means that the generator assigns accurate pixel values to the generated images. However, the L1 loss might not sufficiently illustrate the desired loss for the stress field images. The difference image in Figure 6.5 and Figure 6.10, indicated that generated image is prone to errors in the area around the pit, and errors elsewhere are lower. Consequently, as the loss is the averaged value over the whole image, the loss values became lower than the errors in the pitted area. Using another dimension in the images, where the pit covers more of the image, a larger L1 loss may be yielded. This can consequently reduce the prediction error in maximum SCF and as the generator has steeper gradients to train on when the L1-loss becomes higher. Alternatively, by using a higher λ, the same can be achieved.

The losses from different training session of *Pit2Pix* is shown in Figure 6.11. The best performing network was decided on the training losses. As seen in the figures, the losses for the additional tests are fluctuating, which indicates that the optimized point in the adversarial training is not obtained, and the network has to be trained more. A common problem for cGANs and especially GANs is that they may be unstable during training. The discriminator and generator move around during training to outperform its adversary and can find a point that makes it perform even worse. The L1 losses in Figure 6.11c were used as one important criterion when evaluating the best network. Comparing the loss for the third additional test to the *Pit2Pix* loss made it easy to decide which training network to choose.

Chapter 8

Conclusion and Further Work

8.1 Conclusion

The main objective of this thesis was to investigate the use of machine learning in the remaining life assessment for corroded mooring chains. This can reduce the time consumption compared to today's standard approach as the need for FEA is removed. Furthermore, this can provide cost-saving and a better understanding of the mooring chains' condition.

The use of image-to-image translation by use of a conditional generative adversarial network proved to yield promising results for use in fatigue assessment of corroded mooring chains. The network's generator was able to create real-looking stress field images, by using the pits surface elevation images as an input image. The generator captured the underlying distribution of the training data set.

Two neural networks based in the Pix2Pix architecture, namely *Pit2Pix* and *Pit2PixV1*, were established. The latter introduced a maximum loss to penalize the generator when the maximum value in the generated and true image deviated. However, the applied penalization factor reduced the prediction error in maximum SCF compared to *Pit2Pix* network's prediction. This max loss turned *Pit2PixV1* to underestimate the value more often than the other. Thus, one can conclude that *Pit2Pix* is the best option to go for when assessing the fatigue performance as this is more likely to overestimate the SCF, and thus is conservative.

A finite element model procedure that was able to produce a data set in which the neural networks could use in training, was created. The procedure used an artificial sub surface to obtain the stresses at a critical distance and further transform these stresses into a grayscale image. However, the time used to create the entire data set was high, although using this method provided a good and accurate way to describe the stresses.

8.2 Recommendations for Further Work

Further work would be to create an end-to-end application with the uses of the trained generator. The application should utilize that the position of each pit is known. Hence, the generated stress field image can be mapped back to the original chain. In this way, the end-to-end application can detect the location of the highest stress concentration factor and thus find the critical point to use in the fatigue assessment. The application should also be able to either classify the damage level of the chain based on the generated images or predict the number of cycles before failure based on the highest SCF. The application can, for example, use the procedure presented in section 2.5.

An even larger data set can be made to further strengthen the generator's ability to generate true samples for multiple different pits and sizes. This provides more training samples for the generator and the discriminator to capture the input space distribution. Further, a less time-consuming finite element model procedure should be made. This can be done by either decreasing the number of elements in the meshed model, run multiple analysis in parallel, or utilize other numerical solvers such as ABAQUS.

Further investigations on the actual needed size of the generator and discriminator architecture can be conducted. This will decrease the training and enable a possible end-to-end application to faster generate the stress field images, as a reduction in the number, and size of the layers will decrease the number of variables in the network.

Bibliography

[1] K.-t. Ma, H. Shu, P. Smedley, D. L'Hostis, and A. Duggal. "A Historical Review on Integrity Issues of Permanent Mooring Systems". In: OTC Offshore Technology Conference. OTC-24025-MS. May 2013. DOI: 10.4043/24025-MS.

[2] E. Fontaine, A. Kilner, C. Carra, D. Washington, K. Ma, A. Phadke, D. Laskowski, and G. Kusinski. "Industry Survey of Past Failures, Pre-emptive Replacements and Reported Degradations for Mooring Systems of Floating Production Units". In: OTC Offshore Technology Conference. D041S047R002. May 2014. DOI: 10.4043/25273-MS.

[3] D. A. Baker, Z. Li, S. Wang, X. Zhang, Y. Shao, H. Li, X. Zhan, L. Zhu, and X. (Tao. "Fatigue Assessment of "Corroded" Mooring Chain". In: vol. 3: Structures, Safety, and Reliability. International Conference on Offshore Mechanics and Arctic Engineering. V003T02A068. June 2019. DOI: 10.1115/OMAE2019-96191.

[4] M. Brown, T. Hall, D. Marr, M. English, and R. Snell. "Floating Production Mooring Integrity JIP - Key Findings". In: vol. All Days. OTC Offshore Technology Conference. OTC-17499-MS. May 2005. DOI: 10.4043/17499-MS.

[5] M. Brown, A. Comley, M. Eriksen, I. Williams, P. Smedley, and S. Bhattacharjee. "SS: Mooring System Integrity: Phase 2 Mooring Integrity JIP - Summary of Findings". In: OTC Offshore Technology Conference. OTC-20613-MS. May 2010. DOI: 10.4043/20613-MS.

[6] E. Fontaine, J. Rosen, A. Potts, K.-T. Ma, and R. Melchers. "SCORCH JIP - Feedback on MIC and Pitting Corrosion from Field Recovered Mooring Chain Links". In: vol. Day 4 Thu, May 08, 2014. OTC Offshore Technology Conference. D041S047R004. May 2014. DOI: 10.4043/25234-MS.

[7] J. Rosen, A. Potts, E. Fontaine, K.-T. Ma, R. Chaplin, and W. Storesund. "SCORCH JIP - Feedback from Field Recovered Mooring Wire Ropes". In: vol. Day 4 Thu, May 08, 2014. OTC Offshore Technology Conference. D041S047R005. May 2014. DOI: 10.4043/25282-MS.

[8] E. Fontaine, A. Potts, K.-T. Ma, A. Arredondo, and R. E. Melchers. "SCORCH JIP: Examination and Testing of Severely-Corroded Mooring Chains From West Africa". In: OTC Offshore Technology Conference. OTC-23012-MS. Apr. 2012. DOI: 10.4043/23012-MS.

[9] J. Rosen, G. Farrow, A. Potts, C. Galtry, W. Swedosh, D. Washington, and A. Tovar. "Chain FEARS JIP: Finite Element Analysis of Residual Strength of Degraded Chains". In: vol. Day 2 Wed, October 28, 2015. Offshore Technology Conference Brasil. D021S017R003. Oct. 2015. DOI: 10.4043/26264-MS.

[10] DNV-GL. *Fatigue Design of Offshore Steel Structures, Recommended Practice, DNVGL-RP-C203*. Oct. 2011. URL: https://rules.dnv.com/docs/pdf/DNVPM/codes/docs/2011-10/RP-C203.pdf.

[11] DNV-GL. *Position Mooring Offshore Standard, DNVGL-OS-E301*. July 2018. URL: https://rules.dnv.com/docs/pdf/DNV/OS/2018-07/DNVGL-OS-E301.pdf.

[12] L. Susmel and D. Taylor. "A critical distance/plane method to estimate finite life of notched components under variable amplitude uniaxial/multiaxial fatigue loading". In: *International Journal of Fatigue* 38 (2012), pp. 7–24. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2011.11.015.

[13] D. Ok, Y. Pu, and A. Incecik. "Artificial neural networks and their application to assessment of ultimate strength of plates with pitting corrosion". In: *Ocean Engineering* 34.17 (2007), pp. 2222–2230. ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2007.06.007.

[14] R. Cottis, L. Qing, G. Owen, S. Gartland, I. Helliwell, and M. Turega. "Neural network methods for corrosion data reduction". In: *Materials & Design* 20.4 (1999), pp. 169–178. ISSN: 0261-3069. DOI: 10.1016/S0261-3069(99)00026-6.

[15] E. Fathalla, Y. Tanaka, and K. Maekawa. "Remaining fatigue life assessment of in-service road bridge decks based upon artificial neural networks". In: *Engineering Structures* 171 (2018), pp. 602–616. ISSN: 0141-0296. DOI: 10.1016/j.engstruct.2018.05.122.

[16] Y. Gao and K. M. Mosalam. "Deep Transfer Learning for Image-Based Structural Damage Recognition". In: *Computer-Aided Civil and Infrastructure Engineering* 33.9 (2018), pp. 748–768. DOI: https://doi.org/10.1111/mice.12363.

[17] V. Jaiswal and A. Ruskin. “Mooring Line Failure Detection Using Machine Learning”. In: vol. Day 1 Mon, May 06, 2019. OTC Offshore Technology Conference. D011S014R007. May 2019. DOI: 10.4043/29511-MS.

[18] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. *Pixel Recurrent Neural Networks*. 2016. arXiv: 1601.06759 [cs.CV].

[19] D. P. Kingma and M. Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[20] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[21] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[22] S. Berge and S. K. Ås. *Compendium Fatigue and Fracture Design of Marine Structures*. NTNU, 2017.

[23] P. P. Milella. *Fatigue and Corrosion in Metals*. Springer, 2013.

[24] N. E. Dowling. *Mechanical behavior of materials : engineering methods for deformation, fracture, and fatigue*. eng. 4th ed., international ed. contributions by K. S. Prasad and R. Narayanasamy. Boston, Mass: Pearson Education, 2013. ISBN: 9780131395060.

[25] L. Susmel. *Multiaxial Notch Fatigue, From nominal to local stress/strain quantities*. CRC Press, 2009.

[26] NACE. *Pitting Corrosion*. May 2021. URL: https://www.nace.org/resources/general-resources/corrosion-basics/group-1/pitting-corrosion.

[27] L. Susmel and D. Taylor. “A novel formulation of the theory of critical distances to estimate lifetime of notched components in the medium-cycle fatigue regime”. In: *Fatigue & Fracture of Engineering Materials & Structures* 30.7 (2007), pp. 567–581. DOI: 10.1111/j.1460-2695.2007.01122.x.

[28] A. Holzinger, P. Kieseberg, E. Weippl, and A. M. Tjoa. “Current Advances, Trends and Challenges of Machine Learning and Knowledge Extraction: From Machine Learning to Explainable AI”. In: *Machine Learning and Knowledge Extraction*. Cham: Springer International Publishing, 2018, pp. 1–8. DOI: 10.1007/978-3-319-99740-7_1.

[29] J. Brownlee. *Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation*. Machine Learning Mastery, 2019. URL: https://books.google.no/books?id=YBimDwAAQBAJ.

[30] L. D. Marchi and L. Mitchell. *Hands-On Neural Networks*. Packt Publishing, 2019. ISBN: 978-1-78899-259-6.

[31] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer. "An Introductory Review of Deep Learning for Prediction Models With Big Data". In: *Frontiers in Artificial Intelligence* 3 (2020), p. 4. ISSN: 2624-8212. DOI: 10.3389/frai.2020.00004.

[32] A. Karpathy. "Neural Networks Part 1: Setting up the Architecture". http://neuralnetworksanddeeplearning.com/about.html, Accessed: 12.12.2020. 2020.

[33] S. Vieira, W. H. Pinaya, and A. Mechelli. "Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications". In: *Neuroscience & Biobehavioral Reviews* 74 (2017), pp. 58–75. ISSN: 0149-7634. DOI: 10.1016/j.neubiorev.2017.01.002.

[34] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314. DOI: 10.1007/BF02551274.

[35] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[36] K. Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. ISSN: 0893-6080. DOI: 10.1016/0893-6080(91)90009-T.

[37] M. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/about.html.

[38] M. Hardt, B. Recht, and Y. Singer. "Train faster, generalize better: Stability of stochastic gradient descent". In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. June 2016, pp. 1225–1234. URL: http://proceedings.mlr.press/v48/hardt16.html.

[39] S. Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[40] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[41] Prabhu. *Understanding of Convolutional Neural Network (CNN) — Deep Learning*. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148, Accessed: 12.12.2020. 2018.

[42] A. Vo. *Deep Learning – Computer Vision and Convolutional Neural Networks*. Accessed: 12.12.2020. 2018. URL: https://anhvnn.wordpress.com/2018/02/01/deep-learning-computer-vision-and-convolutional-neural-networks/.

[43] A. Radford, L. Metz, and S. Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[44] M. R. Karim. *Java Deep Learning Projects: Implement 10 Real-World Deep Learning Applications Using Deeplearning4j and Open Source APIs*. Packt Publishing, 2018. ISBN: 178899745X.

[45] A. Karpathy, P. Abbeel, G. Brockman, P. Chen, V. Cheung, R. Duan, I. Goodfellow, D. Kingma, J. Ho, R. Houthooft, T. Salimans, J. Schulman, I. Sutskever, and W. Zaremba. *Generative Models*. Accessed: 21.05.2021. June 2016. URL: https://openai.com/blog/generative-models/.

[46] M. Mirza and S. Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG].

[47] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[48] U. Demir and G. Unal. *Patch-Based Image Inpainting with Generative Adversarial Networks*. 2018. arXiv: 1803.07422 [cs.CV].

[49] Tensorflow community. *Pix2Pix Tutorial*. Accessed: 21.05.2021. Aug. 2019. URL: https://www.tensorflow.org/tutorials/generative/pix2pix.

Appendix A

Generating Image After Training

After training, the generator is aimed to generate images. However, in Tensorflow, one has the ability to choose whether images is to be generated using training=True or training=False. The images presented above in chapter 6 have all been produced using training=True. However, the networks was tested to generate image using training equal false. In Figure A.1 the true maximum SCF is plotted against the predicted maximum SCF.

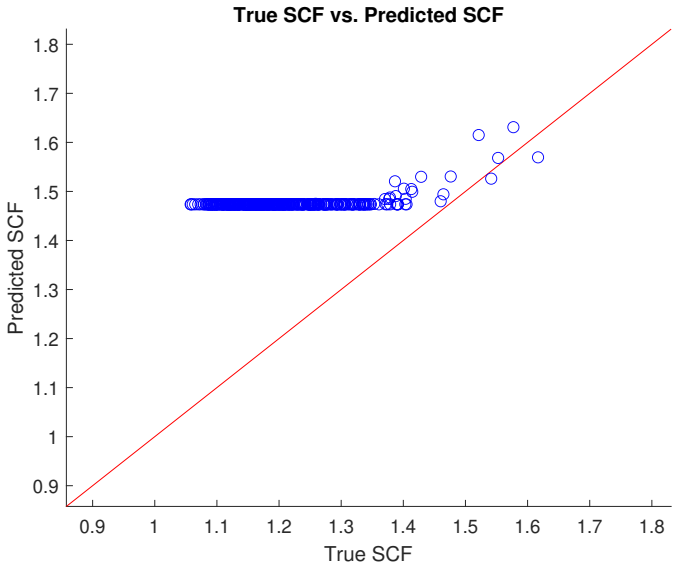


Figure A.1: True maximum SCF vs. Predicted maximum SCF of *Pix2PixV1* with training=False

NTNU

Norwegian University of
Science and Technology