Master Thesis

ABOUT

Self-supervised spectrogram reconstruction using MAE-ViT for anomaly detection in pump audio signals

Rune Drongesen

AALBORG UNIVERSITY

Abstract

This thesis investigates the application of Self-Supervised Learning (SSL) for acoustic anomaly detection in industrial pumps, specifically focusing on identifying pumps with cavitation signatures. A Masked Autoencoder (MAE) framework, utilizing Vision Transformers (ViT), was first pre-trained on data from normal operating pump sound data. This SSL stage aimed to create a model capable of reconstructing a generalized spectrogram of a healthy operating pump. Next for the downstream task of cavitation detection a Convolutional Neural Network (CNN) was fine-tuned. This proposed CNN processed mappings of the pixel-wise absolute difference between the input spectrogram and the MAE-ViT reconstruction, called error maps, to identify anomalous signatures. The performance of this methodology was benchmarked against a Convolutional Autoencoder (CAE), which was trained from scratch on the same dataset. Results showed the MAE-ViT's great ability to reconstruct spectrograms, by achieving a low validation reconstruction loss 0.001 MSE. The fine-tuned CNN, exhibited better anomaly detection capabilities, compared to the CAE. It successfully identified all cavitation anomalies in the validation set, with a Recall of 1 and achieved an AUC score of 0.996. This surpassed the baseline CAE, which recorded a Recall of 0.67 and an AUC score of 0.990. The study concludes that a SSL pre-trained MAE-ViT, combined with a fine-tuned CNN using error maps, offers a more effective framework for detecting cavitation anomalies from acoustic pump data compared to a Convolutional Autoencoder.

Titlepage

Title:
Self-supervised spectrogram reconstruction using MAE-ViT for anomaly detection in pump audio signals.

Theme:
Machine Learning, Self-Supervised Learning, Data processing, Anomaly detection

Project Period:
03/02/2025 - 04/06/2025

Project Group:
1029e

Participants:
Rune Drongesen

Supervisor:
Zheng-Hua Tan

Co-supervisor:
Kevin Wilkinghoff

Grundfos Supervisor:
Nicolai Bæk Thomsen

Date of Completion:
June 3, 2025

AALBORG UNIVERSITY

STUDENT REPORT

Electronic Systems
Institute of Electronic Systems
http://www.aau.dk

Preface

Aalborg University, June 3, 2025

[illegible]

Rune Drongesen
rdrong23@student.aau.dk
112899@grundfos.com

The project is based on a proposal from Grundfos Sound and Vibration Lab. The project proposal encompasses using Machine Learning on pump data, to create a model capable of detecting anomalies or specific errors.

Contents

1 | Introduction

Analyzing and maintaining equipment used for industrial manufacturing is an important aspect for any manufacturing company. The ability to record and interpret acoustic signals from machines is a valuable tool to gain insights into the intricate parts of any machine which cannot be directly monitored by visual inspection or sensors placed directly on the body. These acoustic signals can be used to detect anomalies and prevent breakdowns which can cost a company a lot of time and money.

Using acoustic signals for monitoring production equipment is not the only place where it is valuable. In research and development, these acoustic signals can be used to detect unwanted noise from products that is to be deployed close to humans. Furthermore, it can detect anomalies that might reduce a products lifespan, and other anomalies that needs taken care of before the product reaches the production line.

Moreover, advancements in Machine Learning and Deep Learning have improved the capabilities of the analytical tools by improving accuracy and effectiveness. This advancement enables models to handle and compute vast amounts of acoustic data in order to make more informed decisions on identifying patterns and anomalies. [1] As a result, companies have shifted resources into the development of these Machine Learning tools in order to reduce breakdowns and improve on the final product. [2]

Building on the aforementioned ideas, Grundfos have been collecting acoustic signals of their pumps for the last 12 years. In their quest to be more data-driven, they have created the opportunity to research the option of using Machine Learning for knowledge retention and anomaly detection. They wanted to have a model that was capable of detecting unwanted phenomena such as, degassing, cavitation and structural weakness.

Despite the benefits of using Machine Learning instead of conventional signal processing methods, anomaly detection from acoustic signals still presents several challenges. These include data quantity, data quality, data availability, complexity of acoustics signals and model robustness. Due to the fact that most of the pumps tested at Grundfos are working and in healthy condition the amount of anomalous data is sparse and can contribute to a skewed dataset. For this research, data collection have been done at Grundfos Sound & Vibration Lab (S&V Lab) in their Hemi-anechoic chamber. This controlled environment ensures minimal noise and reduces inconsistencies in the data collection. Providing reliable and great quality data for analysis and decreasing the complexity of the skewed data. Because this research sprouted from ideas after the beginning of data collection, little to no thought had been put into labeling data to use it for further Machine Learning analysis at a later stage. Therefore the annotations attached to each test contains no clear uniformity, which greatly increases the complexity for a Machine Learning model if some data have been labeled incorrectly. To further add to the complexity of the Machine Learning model, the data is collected on a wide variety of pumps giving slightly different acoustic signatures in a spectrogram, making it harder for the model to generalize to what a normal signal should look like.

To try and tackle some of these complexities, this thesis aim to utilize Self-Supervised Learning (SSL) to capture the general spectrogram image of a normal running pump and further fine-tune this model for classification in order to detect any anomalies, as a proof of concept this thesis will focus on detecting the cavitation phenomena. To prove that this approach works, before expanding the model to capture even more anomalies.

The application of Machine Learning for acoustic anomaly detection have been widely studied with various different approaches each with its own advancements. For this introduction some of the related work on SSL and acoustic anomaly detection, that have paved the way for this research will be highlighted below.

Formerly, vision based machine learning mostly relied on supervised CNN, with its great feature extraction capabilities, using a variety of kernels [3] [4]. Later this evolved into using attention for pattern-recognition and image classification[5]. After Transformers have been used in Natural Language Processing (NLP) for some time, this approach saw its breakthrough with Vision Transformers (ViT) [6] for visual based learning, showing that image recognition can be done omitting CNNs and solely using transformers. This later evolved into a combination of the two, creating even better models. Parallel with the transformer evolution the introduction to Self-Supervised Learning was developed with the likes of SimCLR and MoCo [7] [8]. Current trends in vision based machine learning are the integration of SSL with transformers and attention mechanisms.

Müller et al. (2021) [9] propose a novel approach to the field of acoustic anomaly detection by utilizing transfer learning instead of relying on deep autoencoders, the authors extract features from a Mel-spectrogram using pre-trained image classification tasks such as ResNet and AlexNet. These features are then used for anomaly detection using Gaussian mixture Models (GMM) and One Class Support Vector Machines (OC-SVM). This approach showed an improvement compared to the widely used models at that time.

Chen et al. (2022) [10] introduces the concept BEATs (Bidirectional Encoder representation from Audio Transformers), which is a framework for an iterative acoustic pre-training model that optimizes tokens and audio SSL through iterations. Trying to gain a semantic understanding of the audio signal rather than just discrete labels. From the authors experiments the BEATs framework achieves performances on par and even better compared to state-of-the-art audio classification benchmarks. Showing the importance of semantic-rich labels in audio.

Saengthong and Shinozaki (2024) [11] presents GenRep, a method that utilizes a generic feature representation from a robust, pre-trained feature extractor combined with a k-Nearest Neighbors (kNN) for domain specific anomalous sound detection (ASD). This paper manages to combat the understanding that reliable ASD requires extensive labeled data for each target to gain any significant performance. This method outperforms the best Outlier-Exposure based approaches, without any need for labeled data.

Gfeller et al. (2020) [12] Worked on a new Self-Supervised task named Audio2Vec, this task aims to reconstruct slices of a spectrogram from only the past and future slices. In order to create a general purpose audio representation. The study show that when transferring Audio2Vec to different downstream tasks Audio2Vec outperforms fully supervised trained models, when trained from scratch with limited data available. Showing that Self-Supervised tasks are great at reconstructing missing features for later to use this for a classification task.

He et al. (2021) [13] Proposed a new approach for Self-Supervised Learning for computer vision using Masked Autoencoders (MAE). This new approach suggested to mask random patches of an input image and then training the model to reconstruct the missing pixels in order to re-create the original image. This is done using an asymmetric encoder-decoder architecture. While also utilizing the findings of Dosovitskiy et al. (2021) [6] with the 16×16 ViT. From their experiments and tests the MAE model improved accuracy and efficiency of large scale models for transfer learning tasks.

From the related work above this opens the possibility for this thesis to research a MAE framework approach using ViT on a sparse dataset of real-world data and for a further downstream task of fine-tuning a classifier on top to classify anomalies like cavitation in Grundfos pumps.

The structure of this thesis will be as follows. First Chapter 2 will introduce the research question for this thesis, as well as describing the scope. In Chapter 3 the methodology of how the data was collected and what pre-processing steps was taken to prepare the data for analysis. It will explain what implementation approaches was used to detect anomalies in the spectrogram data. Next in Chapter 4 the experiments and results will be presented, explaining how the tests of each model was conducted and how it will be evaluated, the outcome will be compared to the widely used Convolutional Autoencoder (CAE) structure, which acts as the baseline for this thesis. Then the results will be presented with visual representations of the performance. Second to last is Chapter 5 here the findings of this thesis will be discussed, the limitations of this study will also be discussed such as what are the limitations of this model and where it could see improvements. Lastly in Chapter 6 a conclusion will be stated, summarizing the main points of this thesis, providing any concluding remarks on this thesis.

2 | Thesis Scope & Research Question

In this chapter, the thesis scope will be defined. Based on this scope, as well as the introduction in Chapter 1, one or more research questions will be formulated, which will form the basis for this thesis.

2.1 Thesis Scope

As mentioned in Chapter 1 the aim of this thesis is to create a Self-Supervised Learning algorithm that can generalize a spectrogram from a 10 second HATS recording. When this generalized model is created the goal is to fine-tune a model on top that will train on labeled data in a supervised learning model that will classify the data as either normal spectrograms or anomalous spectrograms.

This proposed fine-tuned classification model should be able to out-perform a CAE model that is trained from scratch on the same dataset. Showing the possibilities of using SSL as a pre-trained model with a classifier on top for greater anomaly detection.

This thesis will utilize the approach used in [13] with a Masked Autoencoder (MAE) using Visual transformers (ViT). In order to test if the scalability also works in the reverse direction, because this thesis will use a smaller dataset than introduced in [13]. This is also some of the latest innovations in the field of Self-Supervised Learning on Acoustic data. This approach will be tested against a CAE in order to see if the SSL approach is better than a widely used method for anomaly detection.

Data used in this thesis is collected by Grundfos Sound and Vibration Laboratory in a Hemi-anechoic chamber. Using a Head and Torso Simulator (HATS) located 1 meter from the pump, collecting an acoustic signal. This signal is further processed into a spectrogram to be used in this Machine Learning task.

2.2 Research Question

Based on Chapter 1 and 2 the following research questions were formulated.

1. *"Can a Self-Supervised Machine Learning algorithm, built on a Masked Autoencoder framework, utilizing Vision Transformers, reconstruct a normal representation of a spectrogram of a running pump?"*

2. *"Can a Convolutional Neural Network correctly classify errors in pump data, when used as a downstream task of the pre-trained Self-Supervised Learning model?"*

3. *"Can the Convolutional Neural Network outperform a Convolutional Autoencoder trained on the same data?"*

The reason for choosing a CAE is because it is widely used for anomaly detection [14] and have shown to be effective on video anomaly detection [15][16] and for extracting high level abstract features [17].

3 | Methodology

In this chapter the methodology of all the main parts of this thesis will be described including the raw acoustic signals which were collected and saved before the beginning of this thesis.

3.1 Data Collection

3.1.1 Equipment

The acoustic signals were captured using the Brüel & Kjær (BK) Head And Torso Simulator (HATS) 4100-D [18], this high-precision, low noise system uses a set of $\frac{1}{2}$" Prepolarized Free-field Microphones [19], one positioned in each ear canal of the HATS system. These microphones have a sensitivity of 50 mV/Pa, with frequencies ranging from 6.3 Hz - 20 kHz and a dynamic range of 14.6 dB - 146 dB [19]. When placed in the ear canals of the HATS the two microphones simulate the separation between the human ears and include the interferences introduced by the head and upper body. This ensures an accurate three-dimensional binaural recording [18]. The HATS system is placed at exactly 1 meter from the pump/Device Under Test (DUT). The two signals collected by the HATS are recorded using a BK Type 3160 DAQ [20], before being processed in the BK PULSE Labshop application [21].

3.1.2 Environment

All the recordings made by the HATS system is done in a hemi-anechoic chamber which measures 8 meters in each direction, with the DUT placed in the center of the room free-floating using springs 80 centimeters above ground. The floor is made of concrete, with an epoxy coating on top, in the center of the room is an acoustically isolated platform which is where the DUT is placed, in order to further inhibit outside interference. All of these precautions leads to the chamber having an ambient background noise level of on average 5 dB and creates a great environment to measure sounds without significant noise interference. Which also limits the amount of noise reduction post-processing needed.

3.1.3 Recording Protocol

Each DUT is recorded continuously for around 15 - 60 minutes, depending on the pump. The HATS system records for 10 continuous seconds when the DUT has reached a steady-state, steady-state is reached after the stabilization phase is concluded, here the pump reaches a stable temperature, speed, pressure and flow. The 10 seconds are recorded with a sampling rate of 44.1 kHz and saved as a left and right channel WAV file. All pumps were tested in the same environment.

3.1.4 Data Annotation

Recordings from the chamber have been analyzed by technicians and/or engineers and any inconsistencies in sound power levels, environment, background noise or perceived sound is documented shortly after the recording and analysis. This means that all known cases of cavitation measured, can be found with a string search in the database with all data. If there was any doubt in the annotated data comment the data was manually reviewed to give the correct label. In the end collecting 7190 instances of data, 7095 with no known error and 95 with known cavitation. The cavitation anomalies featured in this dataset, are all collected during sound power tests, with data collection following ISO 3745 [22]. These sound power tests are performed on all types of Grundfos pumps from re-evaluating performance of old pumps to testing R&D projects.

3.2 Data Preprocessing

The sound data from HATS is recorded with a sample rate of 44.1 kHz and saved as a WAV file to utilize the feature that WAV is an uncompressed file type. Making sure all the information collected is saved.

As a result of HATS having bidirectional recording capabilities the saved WAV file have both a right and a left channel. But due to the close resemblance of these channels and the added complexity of two channels. The spectrograms created are only using the right channel. On basis of listening to a small sample of pumps and concluding that no significant difference was apparent between the two channels, and according to the technicians and engineers at Grundfos S&V Lab. They argued that, the right channel would likely pick up more of the potential anomalous sound compared to the left channel, due to the orientation of the pump and the location of the audible error inside the pump. Therefore, the right channel was chosen.

From each of the right channel WAV data, a spectrogram was computed. Using SciPy.signal package [23] the spectrograms were computed with a sample rate of 44.1 kHz on the 10-seconds sound recording. The spectrograms are saved as a 496×496 pixels, PNG image.

The spectrograms computed will be used in both the SSL model, fine-tuned model and the baseline model with the SSL model only using the normal data, and the fine-tuned and baseline model using all collected data. A slight difference in the preprocessing of the data between the models is the labels associated with each data-point. For the SSL the data does not need a specific label to associate it with a specific value, which is also not necessary because the model only uses the normal data. In this case the SSL uses the input data and compare it to the output data in order to calculate a loss. But for both the fine-tuned and baseline model each data-point needs to have a label. These labels have been collected through the database from where the WAV files are stored. The label was determined from the data comments given as a metadata to each test.

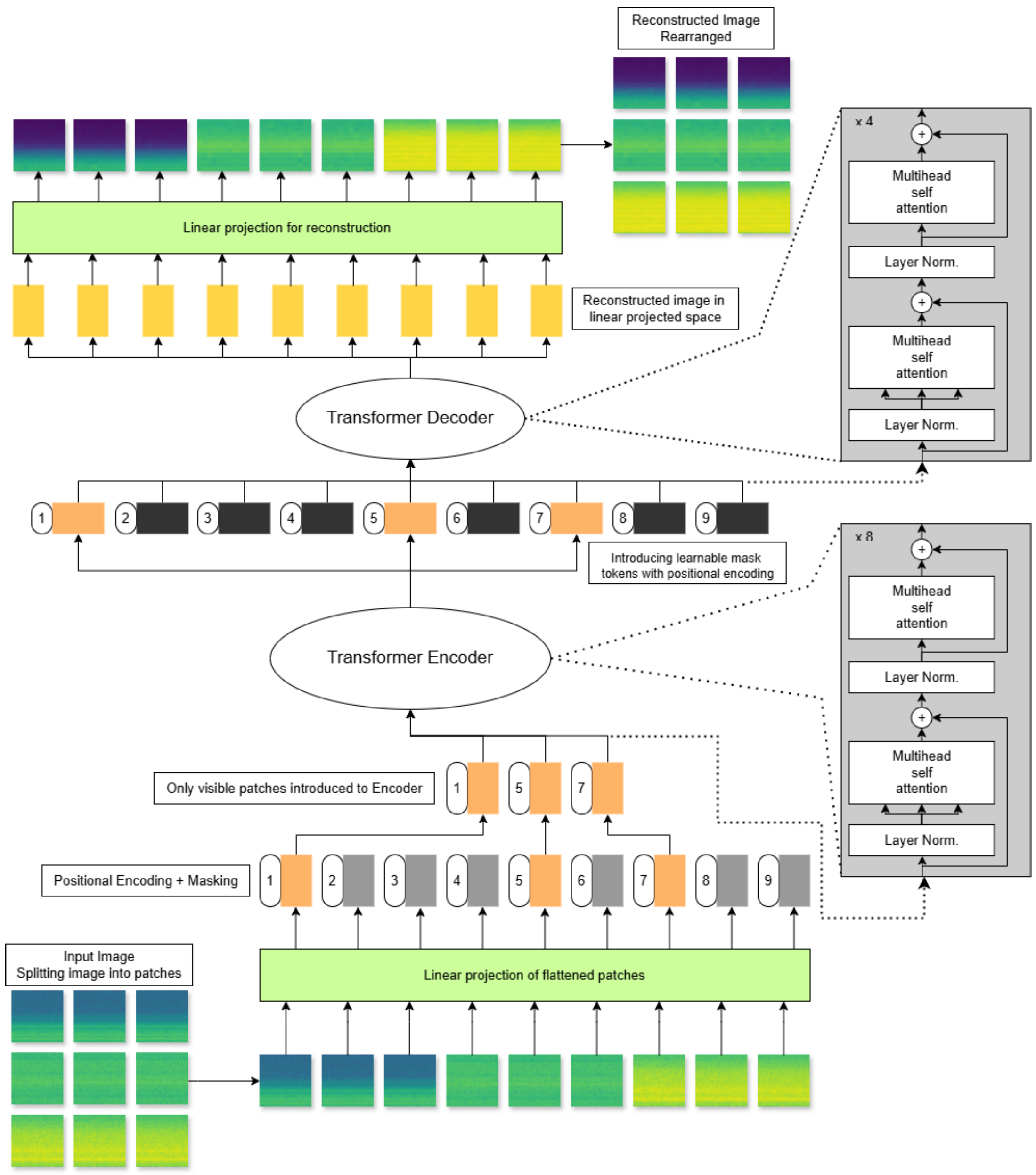
3.3 Self-Supervised Learning

As briefly mentioned in Section 2.1 This thesis will utilize the findings of the research done by He et al. (2021)[13]. Creating a MAE utilizing the ViT encoder and decoder strategy from Dosovitskiy et al. (2021)[6]. In order to create a model capable of recreating a generalized spectrogram image of a working running pump. The downstream task is to create a fine-tuned model on top, capable of detecting anomalous spectrograms by calculating the error between input spectrogram and the reconstructed output spectrogram.

3.3.1 Model Architecture

Visual Transformers

The ViT architecture for this thesis uses the findings from Dosovitskiy et al. (2021) [6]. To deconstruct an image into more manageable sizes before feeding the information into the transformer block.



Reconstructed Image Rearranged
Linear projection for reconstruction
Reconstructed image in linear projected space
Transformer Decoder
x 4
Multihead self attention
Layer Norm.
Introducing learnable mask tokens with positional encoding
Transformer Encoder
x 8
Only visible patches introduced to Encoder
Positional Encoding + Masking
Input Image Splitting image into patches
Linear projection of flattened patches
1
2
3
4
5
6
7
8
9

Figure 3.1: Complete structure of the MAE-ViT model capable of creating reconstructions of generalized normal representations of spectrograms. Starting from patching and masking through the Transformer encoder and decoder.

First major transformation is the patch extraction. For this thesis the input spectrogram image is denoted as $x \in \mathbb{R}^{H \times W \times C}$, where $H = 496$ and $W = 496$, and C is image channels. In this case the images are gray scaled thus making $C = 1$ and therefore can be omitted in the following explanations. The image is partitioned into a regular grid of non-overlapping patches. Each patch with a size of $P \times P$, where $P = 16$ for this model, chosen in line with [13] and [6]. Creating these non-overlapping patches are done by utilizing a convolutional trick. By convolving a kernel of size $P \times P$ with a stride of P. This convolution gives a sequence of $N = \frac{H \times W}{P^2} = 961$ patches of size $P \times P$. This is the sequence length the transformer layers will process.

Following patch extraction is patch flattening. Each patch $x_p \in \mathbb{R}^{P \times P}$, is vectorized to $x_{pf} \in \mathbb{R}^{P^2}$. Which means for this model and with the chosen P the vector length is 256. These flattened vectors are then projected into a D-dimensional embedding space via a trainable linear projection called embedding layer $E \in \mathbb{R}^{P^2 \times D}$. $D = 256$. This gives the model a way of learning meaningful features for each patch token. This results in the initial patch embeddings $z'_p = [x_{p1}E; x_{p2}E; ...; x_{pN}E]$, where $z'_p \in \mathbb{R}^{N \times D}$.

After flattening the patches another critical step is needed in positional encoding. This is critical because the self-attention mechanism used in transformers do not by it self account for the order of the input sequence. Which in images is a fundamental part of understanding the content of an image. Thus making it necessary to incorporate a learnable positional encoding model parameter. By following the findings in [6] a 1D positional encoding creates sufficient performances for image processing tasks. A unique positional encoding vector is learned for each position in the initial input sequence with positional encoding being $E_{pos} \in \mathbb{R}^{N \times D}$ which then becomes $z_0 = z'_p + E_{pos}$, z_0 serves as the first layer of the Transformer encoder.

The central component of ViT architecture is the Transformer encoder, which for this model consists of $L = 8$ identical blocks stacked sequentially. The block structure can be seen in Figure 3.2. The input sequence z_0 is processed iteratively through the L blocks. With the output of block l, being denoted z_l, becomes the input of block $l + 1$. Giving the relation $z_l = Encoder(z_{l-1})$ for $l = 1...L$.

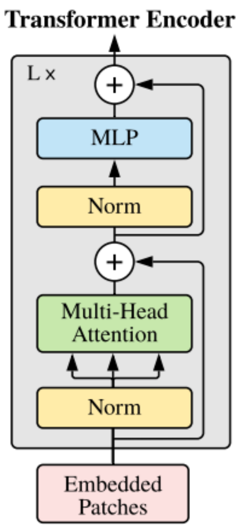


Figure 3.2: Transformer encoder block internal structure. Image from [6]

Each of the encoder blocks consists of two primary components, a Multi-Head Self-Attention (MHSA) mechanism and a Multilayer Perceptron (MLP) network. As extra connections in the encoder block, Layer Normalization (LN) and residual connections are also integrated to improve stability and performance. As shown in Figure 3.2 this encoder follows the pre-normalization variant where LN is applied before each primary component.

MHSA is the fist primary component reached in the encoder. MSHA allows every token in the input sequence z_{l-1} to attend to, and aggregate information from, all other tokens within the same sequence. This creates the ability to capture global dependencies within the sequence. Before the input to MHSA the input sequence is normalized: $z'_{l-1} = LN(z_{l-1})$. This normalized input sequence is then linearly projected into the MHSA to generate Query (Q), Key (K), and Value (V) matrices. These projections are performed independently for each of the attention heads specified for the model in this case $h = 8$. Therefore, for each head $i \in \{1, ..., h\}$, the projections are computed as.

$$Q_i = z'_{l-1} W_Q^i \tag{3.1}$$

$$K_i = z'_{l-1} W_K^i \tag{3.2}$$

$$V_i = z'_{l-1} W_V^i \tag{3.3}$$

Where $W_Q^i \in \mathbb{R}^{D \times (D/h)}$, $W_K^i \in \mathbb{R}^{D \times (D/h)}$, and $W_V^i \in \mathbb{R}^{D \times (D/h)}$ are learned weight matrices specific to each head i. Each head in the MHSA block then computes attention scores using the scaled dot-product attention function [5]:

$$Attention(Q_i, K_i, V_i) = SoftMax\left(\frac{Q_i K_i^T}{\sqrt{D/h}}\right) V_i \tag{3.4}$$

The important part of 3.4 is the dot product calculated in $Q_iK_i^T$. This effectively determines how much attention each token should pay to others, learning the relations in this sequence. This result is scaled by $\frac{1}{\sqrt{D/h}}$ to prevent excessively large dot products, which helps with creating stable gradients during training. The softmax function normalizes the attention weights to sum to one for each query token. Giving the probability distribution for each value vector. This means that the output of each head $head_i = Attention(Q_i, K_i, V_i)$, is a weighted sum of the value vectors. Giving the collected information based on attention scores. The outputs from all h heads are then concatenated along the feature dimension and as the final step passed through a linear projection layer parameterized by $W_O \in \mathbb{R}^{D \times D}$. Giving the final output from the MHSA layer as:

$$MHSA(z'_{l-1}) = Concat(head_1, ..., head_h)W_O \tag{3.5}$$

Before the output from MHSA can be input to the MLP the residual connection from before the first LN layer has to be applied to the output from the MHSA which gives:

$$z''_l = z_{l-1} + MHSA(z'_{l-1}) \tag{3.6}$$

The residual connection used are essential to mitigate the vanishing gradient problem and allowing information form lower levels to propagate more easily.

The second primary component of the Transformer encoder is the MLP. Similarly to the MHSA before the z''_{l-1} token can be input to the MLP it has to undergo a pre-normalization layer giving: $z'''_{l-1} = LN(z''_{l-1})$. The MLP used for this transformer encoder utilizes two layers, where the first layer expands the tokens from dimension D to a larger dimensionality before again in the second layer reducing the dimension back into D again. For this model the first layer increases the dimension to $4D = 1024$. Following the dimensionality expansion comes an activation function, which in this model is the Rectified Linear Unit (ReLU). After the activation function the second layer reduces the dimension from $4D \rightarrow D$. This is the output from the MLP. Before the first Transformer encoder block is finished a second residual connection has to be added to the MLP output. This residual connection comes from before the $LN(z''_{l-1})$ layer used for the MLP input giving the final output of the fist block:

$$z_l = z''_{l-1} + MLP(z'''_{l-1}) \tag{3.7}$$

After all $L = 8$ blocks of the encoder the output sequence is a latent representation of the input image with the shape: $z_l \in \mathbb{R}^{N \times D}$.

Masked Autoencoders

To reduce the computational complexity of the ViT when used on larger images the Masked Autoencoder (MAE) framework is used. By following the findings from He et al. (2021) [13] a masking ratio of 75% is chosen for this model. Thereby reducing the N number of patches to $M_p = \lfloor N \cdot 0.75 \rfloor$, masked patches and $V_p = N - M_p$, visible patches. This reduces the size of the patch embedding to $z'_p = [x_{p1}E; x_{p2}E; ...; x_{pV_p}E$, where $z'_p \in \mathbb{R}^{V_p \times D}$. The change is now z'_p is in the shape $\mathbb{R}^{V_p \times D}$ instead of $\mathbb{R}^{N \times D}$. This by logic then also reduces the positional encoding to the same dimension of $E_{pos} \in \mathbb{R}^{V_p \times D}$.

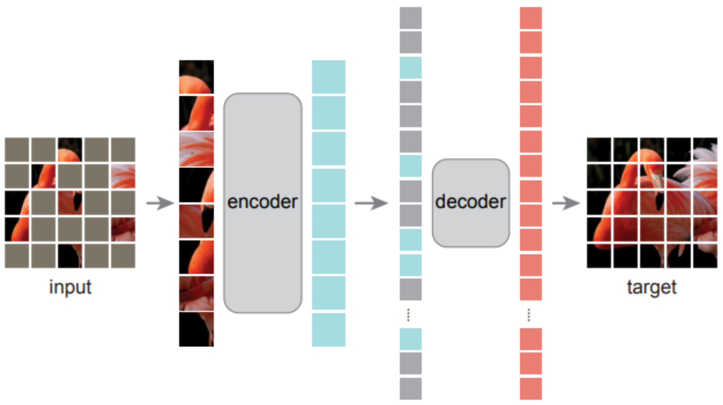


Figure 3.3: General visualization of the MAE framework utilizing ViT. Image from [13].

After the visible patches have been encoded through the Transformer encoder the missing patches are being reintroduced as masked tokens and combined with the encoded visible patches. increasing the size of $z_L \in \mathbb{R}^{N \times D}$. A crucial implementation to be introduced before the combined encoded sequence is fed into the decoder is to introduce positional encoding for the masked tokens, in order for the model to understand the spatial placement of the masked tokens as explained in earlier in Section 3.3.1. To reconstruct the encoded sequence into an image the sequence z_L has to be fed into a decoder. The decoder used for this thesis follows the asymmetric principle from He et al. (2021) [13], where the decoder is substantially smaller than the encoder, more precisely depth of the decoder is $L_d = 4$ and the heads used in MHSA is $h_d = 4$. Other than the reduction of complexity the decoder follows the exact same mathematical methodology as described in earlier in Section 3.3.1. This leaves an important step in the reconstruction process. The output of the decoder is still in the shape $z'_L \in \mathbb{R}^{N \times D}$, and in order to fully reconstruct the image, z'_L firstly has to be reshaped to reach the shape of the patched reconstructed image of $x'_r \in \mathbb{R}^{N \times P^2}$, and finally another reshaping in order to reach $x'_{rp} \in \mathbb{R}^{N \times P \times P}$. Now the image is reconstructed into the full length of 961 patches each of size 16×16. These patches can now be used to train the model. The complete structure of how the MAE framework is utilizing ViT for this thesis can be seen in Figure 3.1

3.3.2 Training Procedure

Self-Supervised Learning Objective

To train the MAE-ViT for image reconstruction purposes the straight forward way would be to compare the reconstructed patches to the original image patches. Though to increase the robustness of the model the reconstruction loss calculated for the model only depends on the reconstructed masked patches. Creating an environment where the model cannot rely on already known patches to pad the reconstruction score. Even though the V_p only consists of 25% of the total patches they could significantly increase the model confidence if included in the loss. All this creates the objective for the model to decrease the reconstruction loss, comparing the reconstructed masked patches to the masked input image patches.

Training Data

As mentioned earlier in 3.1. The data consists of $img = 7190$ spectrogram images with $img_N = 7095$ of these containing no known error. With the knowledge of the downstream task for the model, being to detect anomalies then the main objective for the MAE-VIT model should be to reconstruct the generalized spectrogram of a normal functioning pump. The MAE-ViT is trained only on the 7095 images with no known errors to further emphasize the model role of reconstructing the image of a working pump. The data is partitioned into a training and validation dataset, with a split of 90/10 giving the training dataset $img_T = \lfloor img_N \cdot 0.9 \rfloor$, and the validation dataset $img_V = img_N - img_T$.

To help facilitate model robustness the spectrograms used for training are augmented before it is fed to the model. These augmentations include gray scaling the image, to reduce the image channels and because no meaningful data is collected in the RGB channels as they were arbitrarily constructed when constructing the spectrograms, using a colormap. Lastly the spectrograms have a chance of being flipped both vertically and horizontally both with a 50% chance. This last augmentation is included to force the model to learn relations in the image, because most spectrograms have mostly the same features and look very much alike therefore, changing the orientation of the image forces the model to understand the structure of the spectrogram.

Hyperparameters

The training process for the MAE-ViT model used a set of hyperparameters to define the learning dynamics of the model. The parameters selected are shown in Table 3.1. These values are chosen based on hardware limitations and to ensure effective and good model performance.

Name	**Value**
Batch size	32
Learning rate	0.00001
Weight decay	0.00001
Epochs	200
Mask ratio	0.75
Patch size	16

Table 3.1: Overview of hyperparameter values used for MAE-ViT model.

Batch size of 32 was chosen as a trade-off between the hardware limitations of the machine training the model and to still obtain a good stability in the gradient estimation and generalization across a larger part of the dataset. The hardware used to train and validate all models for this thesis is detailed in Section 4.1.4.

A Learning rate of 1×10^{-5} was chosen based on preliminary tests, which showed this rate provided the most stable result while still achieving a good performance. Increasing the value caused the model to quickly find a local minimum that it was never able to escape. Therefore, never reaching a satisfactory performance. Decreasing the learning rate below the already small value of 1×10^{-5} made the model so cautious that it slowly found a minimum. This minimum was only marginally better than that achieved with a larger learning rate and still inadequate compared to the proposed learning rate. Therefore landing on 1×10^{-5}.

Weight decay of 1×10^{-5} is a rather small value to use in order to prevent overfitting. The penalty it introduces encourages the model to learn smaller weights, in order to slightly improve the models generalization properties. Changes to this value saw little to no change in the model performance and by that logic kept at the first introduced value.

Epochs 200, in comparison with the number of epochs used in [13] is a rather small number. But choosing 200 epochs gave the model sufficient time to optimize the algorithm for decreasing the reconstruction loss as much as possible. As a protection against over-training for the model an early stopping algorithm was implemented. The training and validation performance was monitored throughout each epoch and if the validation set reached a point of convergence, where no improvement was monitored the early stopping algorithm stopped the training process.

A Masking ratio of 0.75 was chosen in accordance with He et al. (2021) [13], which they found encouraged the encoder to learn a holistic and broad interpretation of the dataset. A high masking ratio decreases the training time, by lowering the amount of information the encoder has to interpret. likewise it helps keeping training feasible with the hardware limitations, thus requiring less RAM.

The patch size was kept at $[16 \times 16]$ as found in the base ViT model from Dosovitskiy et al. (2021) [6]. Decreasing the patch size to [8 $\times$ 8] saw a decrease in model performance, and increased the training time as the sequence length handled by the encoder increased from $N = \frac{H \times W}{P^2} = 961$ tokens with $P = 16$, to $N = \frac{H \times W}{P^2} = 3844$ tokens with $P = 8$, and even though the flattened vector length becomes $x_{pf} \in \mathbb{R}^{P^2}$ which for a $P = 8$ becomes 64, should increase the granularity and find finer details in the image, the reconstructed image was worse at generalizing the augmented input image. As for a bigger patch size $P = 32$ was tested and made a slight improvement in testing time, the reconstruction loss saw no reasonable changes, but a visual inspection of the reconstructed spectrogram, showed major decrease in the generalized information portrayed in the reconstruction. Therefore, the patch size was kept at [16 $\times$ 16].

3.3.3 Evaluation Metrics

To asses the effectiveness of the MAE-ViT in the pre-training phase, the quality of the reconstructed image was evaluated. The evaluation of the fine-tuned downstream task will be described in Section 3.4.3. To asses the models main ability to accurately reconstruct a generalized image of the input image based on the visible patches provided from the decoder. The metric Mean Squared Error (MSE) or L2-loss is used for evaluation. MSE is used to measure the average difference between the pixel values in the reconstructed masked patches compared to the original masked pixels.

In the context of this model the MSE can be explained by. Let x'_{mrp} be the reconstructed masked patches and let x_{mp} be the original masked patches. Let M be the set of indices corresponding to the pixels within the masked patches. Then MSE for a single reconstructed patch is calculated as:

$$MSE = \frac{1}{|M|} \sum_{i \in M} (x_{mp_i} - x'_{mrp_i})^2 \tag{3.8}$$

Where $|M|$ is the total number of pixels in a patch. i is the i-th pixel in a patch. Then to calculate the total loss of all masked patches in a reconstructed image the mean of all the patch MSE losses has to be calculated as:

$$Mean_{MSE} = \frac{1}{|M_p|} \sum_{j \in M_p} MSE_j \tag{3.9}$$

Where $|M_p|$ is the total number of masked patches and MSE_j is the MSE of the j-th masked patch. This $Mean_{MSE}$ was computed for both the training and validation dataset. The model weights were updated from the loss computed in the training dataset, but the model is evaluated on the loss computed in the validation dataset.

As an extra form of performance evaluation, a qualitative assessment through visual inspection of the reconstructed image compared to the original image was performed at the end of training. This allows for a subjective evaluation of the models ability to restore image semantics in the overall image. A set of 5 reconstructed images used for visual assessment can be seen in Figure 3.4.

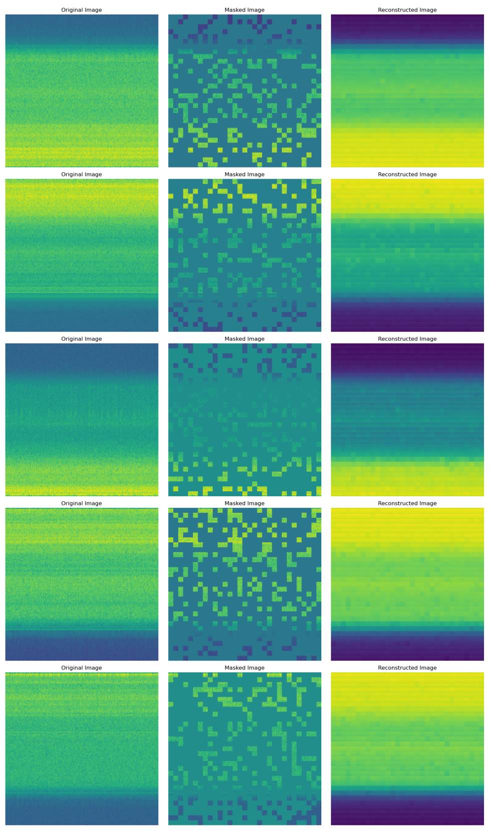


Figure 3.4: A set of 5 reconstructed spectrograms and the original image, used for a subjective visual evaluation of the model performance. Left: Original image. Middle: Masked image. Right: Reconstructed image.

3.4 Anomaly Detection

Following the self-supervised pre-training phase described in Section 3.3, the MAE-ViT model is utilized for a downstream anomaly detection task. This involves fine-tuning a separate, Convolutional Neural Network (CNN) classifier that is placed after the decoder of the MAE-ViT. The core idea is that the pre-trained MAE-ViT, having learned to reconstruct normal pump spectrograms effectively, will exhibit larger reconstruction errors when presented with anomalous spectrograms, as well as exhibiting different structural compositions compared to the reconstruction error of a normal spectrogram. These reconstruction errors, quantified as an "error map", serve as the input to the CNN classifier, which will be trained to distinguish between normal and anomalous conditions based on the patterns within these error maps.

3.4.1 Model Architecture

As described above the fine-tuned model consists of a CNN binary classifier, this classifier will be described later in this chapter. Before explaining the classifier, the error map input should be explained.

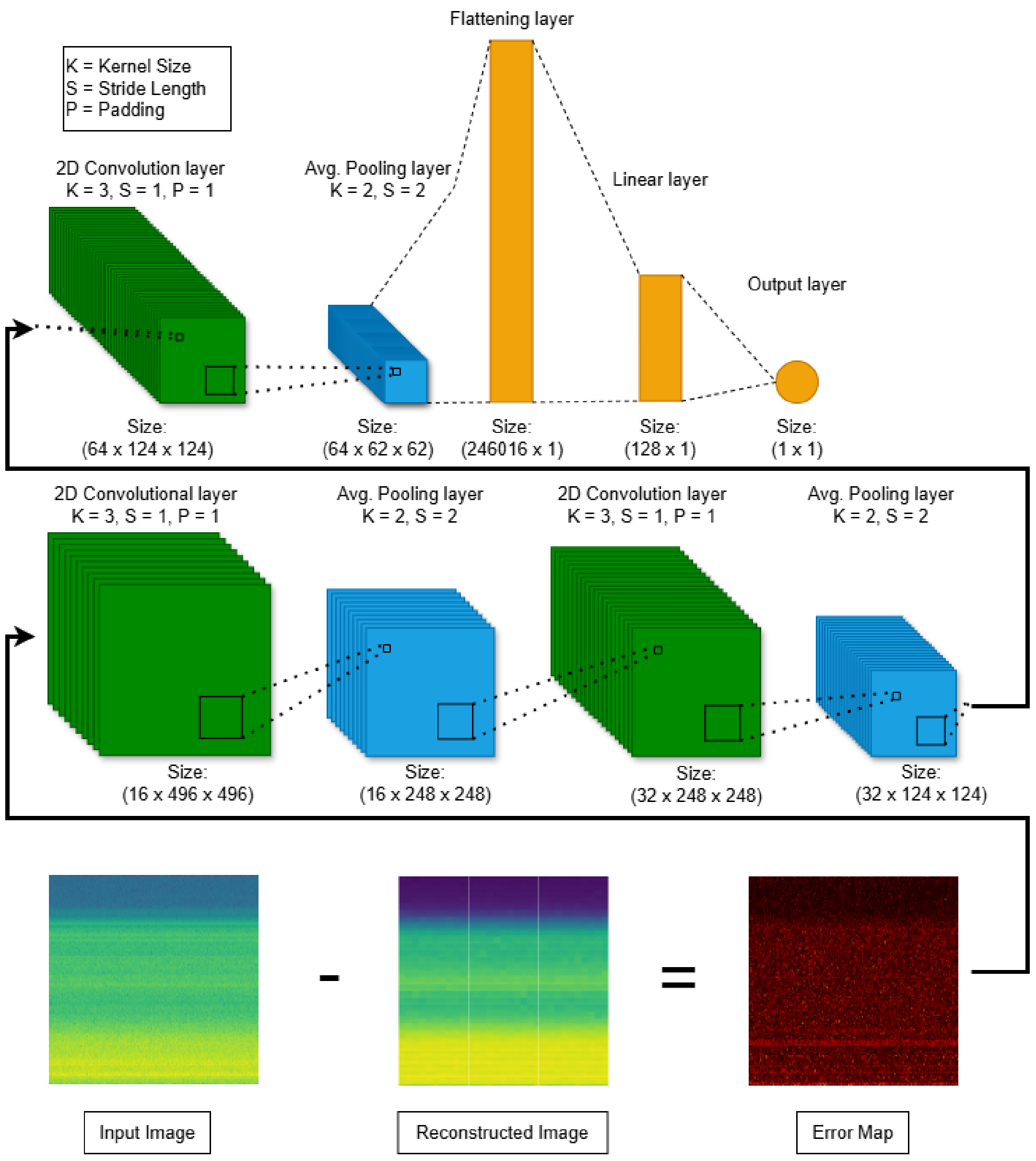


Figure 3.5: Complete structure of the fine-tuned model, from creation of the error map, through the convolutional layers and ending through the fully-connected linear layers at a single output logit.

Error Map

The reconstruction error or error map is derived from the output of the MAE-ViT model, and as described in Section 3.3, this output is $x'_{rp} \in \mathbb{R}^{N \times P \times P}$. Though this form is sufficient, in order to calculate the error map e_{map}, x'_{rp} is linearly transformed into the same dimensions as the input image $x \in \mathbb{R}^{H \times W \times C}$. Such that the reconstructed image becomes $\hat{x} \in \mathbb{R}^{H \times W \times C}$. The error map, that takes the form of $e_{map} \in \mathbb{R}^{H \times W \times 1}$, is calculated as the absolute difference between each pixel value between the original input image and the reconstructed image:

$$e_{map}(i, j) = |x(i, j) - \hat{x}(i, j)| \tag{3.10}$$

(i, j) represents the coordinates for each pixel in the images. Doing this highlights areas where the MAE-ViT model struggles to reconstruct the input, which could indicate deviations from the learned normal pattern. As mentioned above the error map retains the same dimensions as the grayscaled input and reconstructed image and can therefore be directly used as an input to a two-dimensional CNN.

Convolutional Neural Network

The CNN used for this thesis is build upon teaching material in [24]. This Teaching material builds its foundation of knowledge from articles such as LeCun et al. (1989) [25], books from, Yu et al. (2015) [26], and Goodfellow et al. (2016) [27]. A combination of this information gives the building blocks to create a CNN capable of extracting features and classifying images.

The CNN used for this thesis is build with 3 layers of convolutions, 3 corresponding pooling layers and ReLU activation function. All convolution layers utilize a kernel size $K^{(l)} = 3$, a stride length $S^{(l)} = 1$, and a padding $P^{(l)} = 1$. Where (l) corresponds to the l-th convolutional layer, for each convolution layer the feature map Z increases its depth k, the input grayscale feature map is of a single dimension, after each convolution k increases from $1 \rightarrow 16 \rightarrow 32 \rightarrow 64$. The two-dimensional kernel $W_k^{(l)} \in \mathbb{R}^{K^{(l)} \times K^{(l)}}$ is convolved across the input image. For each k and l a bias term $b_k^{(l)}$ is added. From this the output value before activation becomes:

$$Z_k^{(l)} = b_k^{(l)} + W_k^{(l)} * e_{map}^{(l)} \tag{3.11}$$

Equation 3.11 is adapted from [28].

After convolution $Z_k^{(l)}$ becomes the input for the activation function. This ReLU activation function decides how important each part of the feature map is, this operation is done element-wise, on each of the values in $Z_k^{(l)}$ and for all dimensions k, like:

$$Y_k^{(l)} = ReLU(Z_k^{(l)}) = \max\left(0, Z_k^{(l)}\right) \tag{3.12}$$

Equation 3.12 is adapted from [29].

As for now the output of the activation function is still at the same dimension as the input $Y^{(l)} \in \mathbb{R}^{H \times W \times 1}$. To further hone the feature map to the most important features an average pooling strategy is adapted. This average pooling has a kernel size $K_p^{(l)} = 2$ and a stride length of $S_p^{(l)} = 2$, this creates a sliding window that calculates the average value encompassed by the average pooling kernel $K_p^{(l)}$ across the feature map and therefore halves the spatial dimensions of the input feature map. The final feature map for layer l can be calculated by:

$$F_{k,i,j}^{(l)} = \frac{1}{K_{pH}^{(l)} \cdot K_{pW}^{(l)}} \sum_{x=0}^{K_{pH}^{(l)}-1} \sum_{y=0}^{K_{pW}^{(l)}-1} Y_{k,i \cdot S_p^{(l)}+x, j \cdot S_p^{(l)}+y}^{(l)} \tag{3.13}$$

Equation for average pooling 3.13 is adapted to fit this use case from [30].
K_{pH} and K_{pW} are the height and width of the average pooling kernel. Now the output of the first convolution block after all steps halves the dimension to $F_k^{(l)} \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times k}$

This procedure is done 3 times with each time increasing feature map depth dimension but decreasing the spacial dimension. After 3 convolution blocks the output dimensions for the feature map becomes $F^{(3)} \in \mathbb{R}^{\frac{H}{2^3} \times \frac{W}{2^3} \times 64}$. In other words creating 64 feature maps each with 62×62 features. These features can now be further fed into a two-layer fully-connected neural network.

Multilayer Perceptron

The final two layers of the CNN converts the three-dimensional feature maps into a single feature vector containing all the information and reducing the vector dimension to output a single logit used to make a binary classification prediction.

This is achieved by firstly vectorizing $F^{(3)} \in \mathbb{R}^{\frac{H}{2^3} \times \frac{W}{2^3} \times 64}$ to obtain a shape of $x_{linear} \in \mathbb{R}^{\frac{H}{2^3} \cdot \frac{W}{2^3} \cdot k} = \mathbb{R}^{62 \cdot 62 \cdot 64} = \mathbb{R}^{246016}$, this flattened feature map is then fed into a fully-connected linear layer with ReLU as activation function to reduce the feature neurons from 246016 to 128 neurons. Next fully-connected layer converts the 128 neurons into a single output logit neuron. The full structure of the fine-tuned model can be seen in Figure 3.5. This logit can now be used to train the fine-tuned model to differentiate between normal and anomalous data.

Baseline Model

In order to compare the fine-tuned model performance to a model that is widely used for anomaly detection, a CAE model has been implemented to give a baseline performance. The proposed anomaly detection fine-tuned model should beat the baseline model in a direct comparison.

The CAE created for this comparison is comprised of 4 convolutional layers encoding the input image. The 4 layers use a kernel size of $K_b^{(l)} = 3$, a stride length of $S_b^{(l)} = 2$ and a padding of $P_b^{(l)} = 1$. This means, compared to the convolutional layers of the fine-tuned model, the baseline CAE does not need a pooling layer in order to decrease the feature maps spatial dimensions while still increasing its depth. This is achieved by $S_b^{(l)} = 2$ skipping the convolutional calculation to only be every other grid value in both horizontal and vertical direction, this stride hopping to reduce the feature map size is depicted in Figure 3.6.

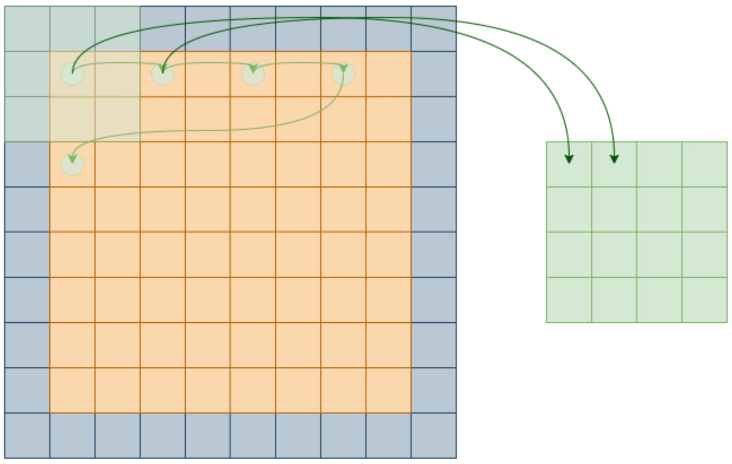


Figure 3.6: Convolutional strides, reducing feature map size. Orange is input feature map. Gray is padding. Green opaque square is kernel, green circle is center of kernel for each convolution calculation. Green square on the right is the output feature map.

Following close to the same methodology of the CNN-based binary classifier of the proposed system. The baseline CAE calculates the feature map $Z_{b_{k,i,j}}^{(l)}$ as in Equation 3.11, and also uses ReLU as activation function as in Equation 3.12. But then skips the pooling function. Therefore, after 4 layers of convolution and activation the feature map depth k_b has gone from $1 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 128$, such that the latent space feature map $Y_b^{(4)}$ is of the dimensions $Y_b^{(4)} \in \mathbb{R}^{31\times31\times128}$. This encoded latent space is used for classification, when combined with a shallow MLP just like in the fine-tuned model case. The key difference in these two approaches lies in the decoder trying to recreate the input image from the latent space representation. As a difference from the pre-trained model decoder, this decoder is symmetric with the encoder and utilizes deconvolution instead of transformers, this also means that the kernel size, stride length and padding are the same as for the encoder. The deconvolution decrease the depth of feature maps k_b at the same rate the convolution increased the depth going from $128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 1$, giving the reconstructed image when $k_b = 1$.

The shallow MLP used for classification in the baseline model has the same structure as that of the fine-tuned model. More accurately it starts by flattening the encoded latent space by vectorization, making $Y_b^{(4)} \in \mathbb{R}^{31\times31\times128} = \mathbb{R}^{123008}$ then input the vectorized features into a linear layer transforming the features down to 128 neurons and after using the ReLU activation function, a dropout chance of turning off each neuron is implemented to better generalize the feature transformation. After dropout a further linear transformation, transforms the neurons down to a single binary logit. The full structure of the baseline model can be found in Figure 3.7.

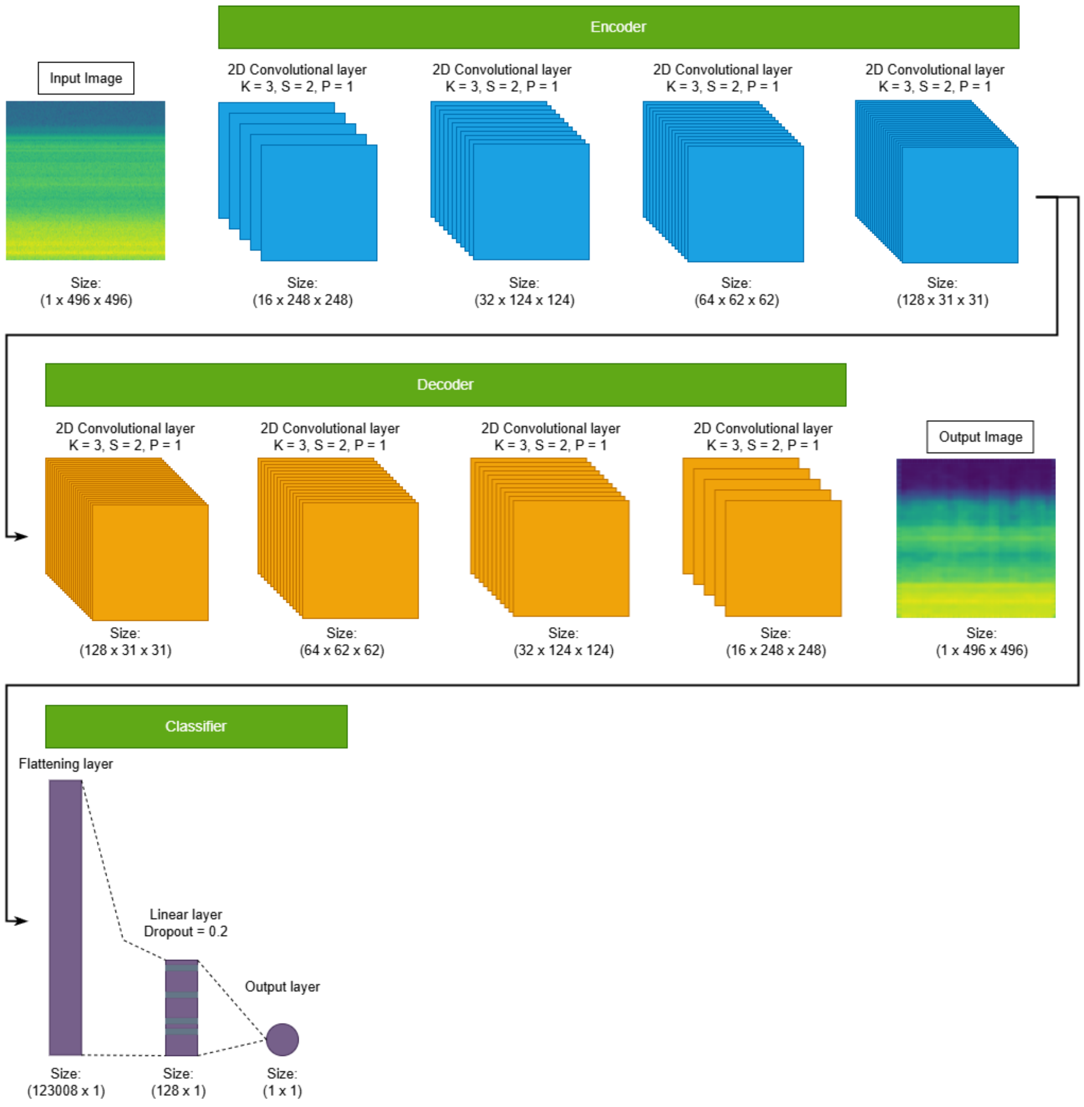


Figure 3.7: Full structure of the baseline model, with encoder, decoder and classifier. The gray lines in the linear layers symbolizes a dropout of 0.2.

These methodologies for both the fine-tuned model and the baseline model, now makes it possible to train the fine-tuned model to classify anomalies and for the baseline model to train to both reconstruct images and classify anomalies.

3.4.2 Training Procedure

Classification Learning Objective

To train the fine-tuned model for classification it utilizes the fact that this is a binary classification problem and therefore uses Binary Cross Entropy (BCE) loss to calculate the loss for each input and uses this to update the weights of the model. During the training of the fine-tuned model, the weights of the pre-trained MAE-ViT model are frozen and are therefore not affected by the training done to the fine-tuned model.

BCE, or in this case Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) [31], is well-suited for binary classification, as it measures the difference between the predicted probabilities and the actual binary labels. To be able to use the logits for a probability score, the function deploys a sigmoid function layer that converts the logits into a probability distribution, and then calculates the BCE.

For the fine-tuned model, let y_n be the true label for the n-th spectrogram, where $y_n = 1$ for anomalous and $y_n = 0$ for normal, and let x_n be the raw logit output of the final linear layer of the model. Then by firstly applying the sigmoid function to the logit the probability becomes:

$$p_n = \sigma(x_n) = \frac{1}{(1 + e^{-x_n})} \tag{3.14}$$

Then, for a single spectrogram the BCE Loss is calculated as:

$$BCE_n = -[y_n \cdot \log(p_n) + (1 - y_n) \cdot \log(1 - p_n)] \tag{3.15}$$

The total loss for a single batch used to update model weights is the mean of all the individual BCE losses for that batch. While the BCE loss was monitored for both the training and validation dataset, the weight for the model was only updated based on the training loss. But evaluated on the validation loss.

For the baseline model, it also utilizes the binary nature of this problem and uses BCE-WithLogitsLoss for the classification, but as an extra constraint to the loss function, the MSE Loss is calculated for the reconstruction by the decoder. Firstly, it is done by calculating the MSE loss for the whole image, let N correspond to the pixels within the image, and let S be the input spectrogram and $\hat{S}$ be the reconstructed spectrogram, then MSE is calculated by:

$$MSE = \frac{1}{|N|} \sum_{i \in N} (S_i - S)^2 \tag{3.16}$$

Then to calculate the total loss for the baseline model the MSE calculation is added to the BCE loss calculation from Equation 3.15 to then become:

$$L_n = MSE_n + BCE_n \tag{3.17}$$

Where n corresponds to the n-th spectrogram in the given batch. Where it is, in the end, averaged across the whole batch before it is used to update model weights. This loss is as with the other models calculated for both the training dataset and the validation dataset, where the training loss is used for updating weights and the validation loss is used for evaluating the model. This combined loss creates the value for updating the weights throughout the baseline autoencoder.

Training Data

As the fine-tuned model purpose is to classify anomalies the complete dataset comprised of the full $img = 7190$ images mentioned in Chapter 3.1 with both the normal data and the anomalous data is used for training this model. As with the MAE-ViT model the dataset is also partitioned into a training and validation dataset, and again as in Chapter 3.3.2 the split is 90/10, giving training dataset $img_T = \lfloor img \cdot 0.9 \rfloor$, and the validation dataset $img_V = img - img_T$.

Following the methodology of the MAE-ViT again, the dataset used for training, augments the images before training, this includes grayscaling, and flipping the images both in the horizontal and vertical direction, both with a 50% chance.

The baseline model uses the same full dataset with both normal and anomalous images, as well as the 90/10 training, validation split. Furthermore, it also uses the same augmentations as the fine-tuned model.

Hyperparameters

To train the fine-tuned model a set of hyperparameters were chosen to define the learning dynamics of the classification model. The parameters used are shown in Table 3.2. These values are chosen to increase performance of the model while still being limited due to hardware.

Name	**Value**
Batch Size	32
Learning rate	0.00005
Weight decay	0.00001
Epochs	200

Table 3.2: Overview of hyperparameter values used for fine-tuned model.

A walkthrough on why the different values were chosen, can be found in Chapter 3.3.2. The only difference in the hyperparameters are the learning rate. For the fine-tuned model the learning rate was increased to 5×10^{-5}, this slight increase in value, was found during preliminary testing and showed a more robust convergence, consistently converging to the same loss value on par with lower and usually more stable learning rates. Otherwise the relevant hyperparameters did not change from the pre-trained MAE-ViT model to the fine-tuned model.

To train the baseline model a similar set of hyperparameters were chosen to again try and manipulate the dynamics of the model in a favorable way. The chosen hyperparameters and values can be found in Table 3.3.

Name	**Value**
Batch Size	32
Learning rate	0.00001
Weight decay	0.00001
Epochs	200
Dropout	0.2

Table 3.3: Overview of hyperparameter values used for the baseline model

Seeing that all parameters that are the same both in this model and the MAE-ViT model, the explanation for each of the parameters can be found in Section 3.3.2. Compared to both MAE-ViT and the fine-tuned model, the hyperparameter dropout is added. It was shortly described in Section 3.4.1. Dropout is a method often used to help models generalize its connections better and therefore get a broader understanding of the dataset. With a dropout of 0.2, the model has a probability of 0.2 for each of the 128 neurons, to turn off the activated neuron and convey no information to the next coming layer, and this happens for every single item processed by the model. The value of 0.2 helps the model generalize incoming information ensuring no channel through the model conveys all the information. By preliminary tests values above 0.2 seemed to turn of too many channels for the model to find any coherence in the information conveyed. Decreasing the dropout to 0.1, seemed to have no particular effect on the performance compared to 0.2. But turning dropout off and setting it to 0.0 increased the information to the single output logit such that it was prone to overfitting and never finding the scattered anomalies.

3.4.3 Evaluation Metrics

To evaluate the performance of the fine-tuned model on the downstream task of binary classification, several metrics were employed. These metrics provide a deeper understanding of the model's ability to differentiate and classify the spectrogram data. While the training process utilized BCEWithLogitsLoss [31], with positive weighting to update the model weights, a confusion matrix and Receiver Operating Characteristics - Area Under the Curve (ROC-AUC) were used for a more thorough performance assessment.

After training, where the lowest validation loss was found, a more detailed assessment of the classification performance was done. This assessment utilizes a confusion matrix. The confusion matrix provides a summary of the models predictions on the validation dataset, breaking the guesses down into correct and incorrect classifications. An example of a typical confusion matrix generated for this project can be seen in Figure 3.8.

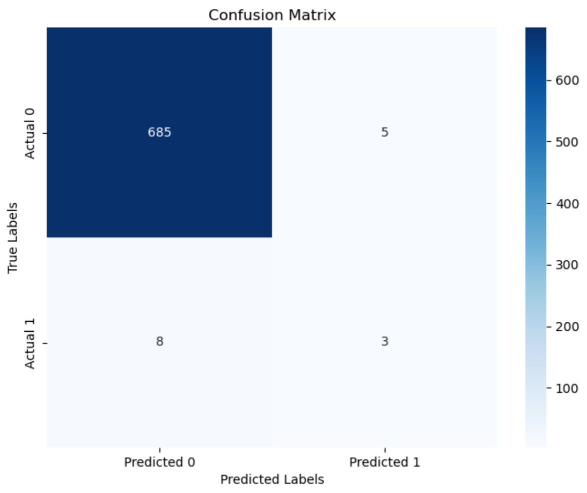


Figure 3.8: Confusion matrix used for assessing performance, with entries in all four key values, 0 = Normal, 1 = Anomaly.

The confusion matrix consists of four key values, and in this case let anomalous data be the positive class, labeled 1, and normal data be the negative class, labeled 0, using Figure 3.8 as an example, then the key values become.

- True Positive (TP): The number of correctly classified anomalous data, in this case the value is 3.
- True Negative (TN): The number of correctly classified normal data, in this case the value is 685.
- False Positive (FP): The number of normal data incorrectly classified as anomalous data, is in some cases also called Type I error, in this case the value is 5.
- False Negative (FN): The number of anomalous data incorrectly classified as normal data, in some cases called Type II error, in this case the value is 8.

From these four key values several of the standard performance metrics can be calculated, such as Accuracy, Precision, Recall and F1-Score. Due to the skewness of the data distribution of this project, the accuracy score is a misleading performance indicator. Because even if the model predicted all negative (all normal data) the model would still reach an accuracy of 97.7% accuracy, while the number is high the outcome is not desirable. Therefore, Precision and Recall are a much better descriptor of the model performance. Furthermore, while F1-Score is a harmonic mean between Precision and Recall, the score gives equal importance to both Precision and Recall [32], which is not desirable in the circumstance of this thesis. Because for use case of this thesis which is based on a specific problem Grundfos wanted to solve. The importance of reaching a Recall of 1 is much more important than reaching a Precision of 1, because a malfunctioning pump, can be costly when it fails and has to be repaired, more in-depth reasoning behind this statement is elaborated in Chapter 5. Recall is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{3.18}$$

This describes the True Positive Rate (TPR) which measures the models ability to find anomalies. Whereas Precision or Positive Predictive Value (PPV) measures the proportion of positive instances that were correctly identified, given as:

$$Precision = \frac{TP}{TP + FP} \tag{3.19}$$

Finally the ROC-AUC is used to evaluate the model's discriminative power across all possible classification thresholds, and is thus a threshold-independent metric. The ROC curve plots the Recall against the False Positive Rate (FPR) $FRP = \frac{FP}{FP+TN}$, at various threshold settings. AUC give a single value score summarizing the performance score across all thresholds, where AUC of 1 represents a perfect classifier and AUC of 0.5 indicates performance equal to random guessing. This means a higher AUC scores signifies a better ability for the model to distinguish between normal and anomalous data.

The relevant metrics, used for performance assessment were all computed on the validation dataset. BCE loss curves were computed for both the training and validation set, in order to evaluate if the fine-tuned or baseline models were overfitting to the training data, and thereby sacrificing performance on the validation data. These evaluation metrics are sufficient to make a quantitative assessment of both models performance, in order to determine if the fine-tuned model is performing better than the comparison model.

4 | Experimental Results

As a brief reiteration of the goals for the experiments conducted in this thesis as stated in the research question, Chapter 2 and explained further in Sections 3.3 and 3.4. The goal of the MAE-ViT pre-trained model, was to create a model capable of learning a normal representation of a spectrogram created from a 10-second audio recording of a pump and from that, able to reconstruct it. The experiments conducted on the MAE-ViT model consisted of training the model and validating the performance.

The goal for the Anomaly detection model was to utilize the pre-trained MAE-ViT model and fine-tune a downstream task with extra layers making the anomaly model capable of detecting and classifying anomalous spectrograms. The experiments conducted, therefore, consists of training the fine-tuned model, validating the performance and to compare the performance of the fine-tuned model to the baseline CAE model to see if the extra computing power generates better performance.

4.1 Experimental Setup

In experimental setup a short recap of all the methods used for each model will be described, and to further specify values used for each method and model. The broad methodology explanation can be found in Chapter 3.

4.1.1 Datasets

The data used for this thesis is collected at Grundfos S&V Lab, in their Hemi-anechoic chamber. Using a HATS system to collect 10-seconds sound recordings of a DUT. The collected data used for this thesis consists of saved recordings done in the laboratory on most types of Grundfos pumps in the last 5 years. The recordings are converted into spectrograms as presented in Section 3.2, these spectrograms are used as a two-dimensional input for the models.

The complete dataset consists of 7190 spectrograms, with 95 of these spectrograms having known anomalies, making it 7095 normal spectrograms. For the MAE-ViT model the aim was to reconstruct the general structure of a normal spectrogram, this is why experiments on this model was only conducted using the 7095 normal spectrograms. With a data split of 90/10, making training dataset 6385 entries and validation dataset 710 entries.

Both the fine-tuned classification model and baseline CAE model, utilizes the whole dataset of 7190 spectrograms. These are also both in a 90/10 training and validation split, giving training 6475 entries and validation 720 entries.

4.1.2 Model Configurations

MAE-ViT Pre-training Model

Some of the key model configurations used for the MAE-ViT pre-training model, which is described in greater detail in Section 3.3.1, is that the image with size $[496 \times 496]$ pixels is divided into $[16 \times 16]$ patches and the MAE framework with a masking ratio of 0.75 creates 241 visible patches which is fed into the encoder in the ViT. The encoder of the ViT has a depth of 8 and uses 8 heads in the MHSA mechanism, and the MLP has a hidden layer of $256 \cdot 4 = 1024$ neurons. While the decoder of the ViT is asymmetric to the encoder and is therefore down-scaled to a depth of 4 and 4 heads and with the same neurons in the hidden layer of the MLP. But the input of the decoder consists of both the encoded visible tokens but also the missing masked tokens, that the decoder should be able to reconstruct. The full overview of the MAE ViT model structure can be seen in Figure 3.1.

Fine-tuned Downstream Classifier

The fine-tuned classifier utilizes a CNN structure in order to distinguish between normal and anomalous data. The input to this CNN classifier is the error map created from subtracting the reconstructed image from the original image in order to find where the reconstruction differs. This error map is fed into a 3-layer CNN model, utilizing convolution of a $[3 \times 3]$ kernel, an average pooling layer with a kernel of $[2 \times 2]$, and a ReLU activation function. Ending the CNN is a flattening layer before a single hidden layer, and an output layer outputting a single logit for classification. This fine-tuning model structure can be seen in Figure 3.5.

Baseline Model

The baseline model is a CAE, with a symmetric encoder and decoder structure, both utilizing 4 CNN layers. The encoder takes the $[496 \times 496]$ input image and encodes it into a latent space with the dimension $[128 \times 31 \times 31]$, by convolving the image with a kernel of size $[3 \times 3]$ and a stride length of 2 and padding of 1. From the latent space the decoder reconstructs the image. Together with the decoder, a shallow MLP, which is placed after the encoder, classifies the image by vectorizing the latent space features into a flattened representation, then feeding it into a hidden layer of 128 neurons with a dropout of 0.2, and then to an output layer of a single classifying logit. The more in-depth description of both the fine-tuned classifier and the baseline model can be found in Section 3.3.1. The full visualization of the baseline model can be seen in Figure 3.7.

4.1.3 Training Regimes

Training MAE-ViT

The aim for training the MAE-ViT model is to create an algorithm that is capable of reconstructing the general normal spectrogram structure of a working pump. This training is done by utilizing MSE loss, in order for the weights of the model to be updated by the reconstruction error. This reconstruction error is explained in Section 3.3.3. All the hyperparameters used for training this model can be found in Table 3.1, the most important parameters for training this model is, learning rate of 1×10^{-5}, batch size of 32, and a patch size of 16. While the epochs are also a key hyperparameter, the value was set high enough that either early stopping would prevent the model from overfitting or the model saw a clear convergence such that the model would not benefit from training further. All these parameters were chosen to improve the performance of the model.

Training Fine-tuned Model

The training outcome from the fine-tuned model should be a model capable of doing anomaly detection from error maps. This binary classifier should be capable of telling the difference between a normal functioning pump and one with abnormal behavior. The binary classification training is done by using BCE loss to train the weights of the fine-tuned model. The weights of the MAE-ViT pre-trained model, which the fine-tuning model is built upon, is frozen, such that it is only the weights of the fine-tuned model that is being updated by the BCE loss. The use of BCE loss is described in further detail in Section 3.4.3. The hyperparameters used for the fine-tuned model can be seen in Table 3.2. The key parameters here are the batch size of 32 and the learning rate of 5×10^{-5}, and again the epochs have been set at 200 such that the early stopping has time to step in if the model overfits and long enough to ensure stable convergence.

Training Baseline Model

The baseline model is used to set a good performance of anomaly detection for the fine-tuned model to try and outperform. Which means that the baseline model is also a binary classifier capable of detecting anomalous data. Different from the fine-tuned model the baseline model uses both MSE loss and BCE loss in order to train the weights of the CAE. It uses BCE as the classification loss and then uses MSE as the reconstruction loss and the combination of these two losses updates the weights of the CAE. The hyperparameters of the baseline model are very similar to those of the fine-tuned model and can be found in Table 3.3. The key parameters are batch size of 32, learning rate of 1×10^{-5}, and then dropout of 0.2. Again the epoch parameter is set to 200 as the two other models, by the same logic.

4.1.4 Implementation Details

For implementing the models and the training regimes, everything was built in Python (3.12.4) [33]. Either PyTorch (2.5.1) [34] or PyTorch Lightning (2.5.1) [35] was used when creating models and structuring training and datasets. For evaluation metrics and plotting, a few different libraries were utilized, Scikit-Learn (1.6.1) [36] was used for metrics such as ROC and confusion matrices. NumPy (2.0.1) [37] was used for vector calculations on the CPU. Matplotlib (3.10.0) [38] and Seaborn (0.13.2) [39] was used for plotting the visualization of different metrics used throughout this thesis.

The hardware used for training is a HP ZBook Fury 16 G10. This laptop has a 13th Gen Intel Core i7-13850HX, with 20 total cores, 28 threads and a turbo speed of 5.30 GHz. It uses a NVIDIA RTX 2000 Ada Generation laptop GPU, with 8 GB GDDR6 VRAM and with 3072 CUDA cores. The CUDA cores in the GPU enables the use of easy parallelization, reducing calculation times drastically. This creates the basis for the experiments conducted for this thesis.

4.2 MAE-ViT Pre-training Results

The MAE-ViT pre-training model was trained for 200 epochs with a learning rate of 1×10^{-5}, a batch size of 32, and a mask ratio of 0.75, as listed in Table 3.1.

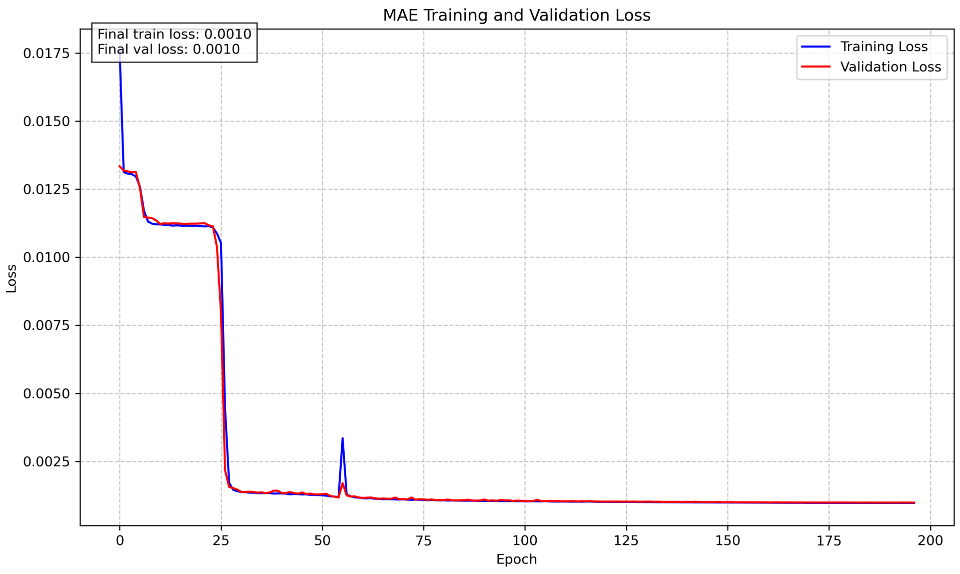


Figure 4.1: Training and validation loss curves for the best performing MAE-ViT model.

After preliminary testing of hyperparameter variations, the loss curves of the best

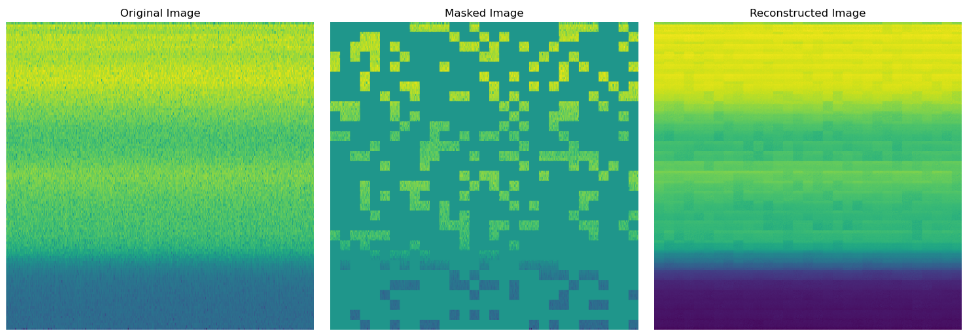


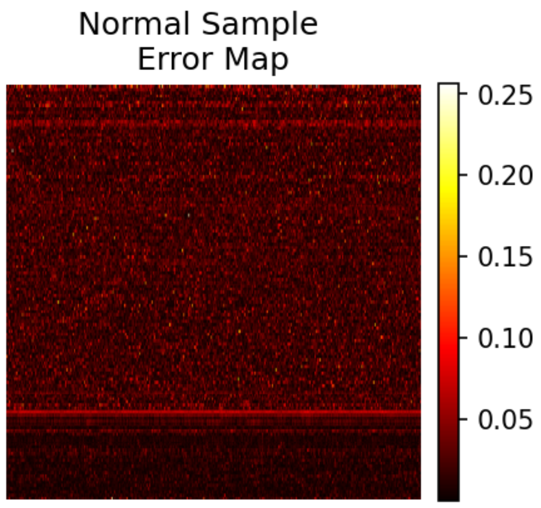
Figure 4.2: Figure depicting a reconstructed image of the MAE-ViT model. Left: Original input image. Middle: Masked image showing the visible patches fed to the encoder. Right: Reconstructed image.

performing model, with the stated hyperparameters gave results as seen in Figure 4.1. The loss curves exhibit behaviors indicating a successful training of a model. Both training and validation loss shows a clear downwards trending behavior and in the end converges to a stable MSE reconstruction loss off 0.001. Having the validation loss closely tracking the training loss throughout the process indicates that the model learned representations that generalize well for the validation set, with minimal overfitting.

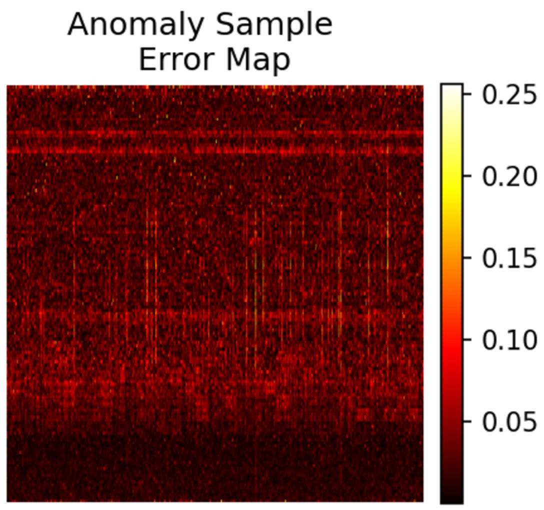
During training a few notable events can be seen in the loss curve. From the initial epochs, the training loss sees a rapid descent in loss, which is to be expected when weights are randomly initialized. But rather quickly a very small plateau is reached before rapidly decreasing again. These initial drops suggests that the model has learned some very basic information about the structure of the data. After the initial learning a new larger plateau is reached which spans around 20 epochs, where a little to no improvement happens. But suddenly a significant drop in loss occurred indicating the model learning some more defining features of the data. After this the models enters a stable convergence phase where only minor adjustments happens.

Although at around 55 epochs a sudden transient fluctuation primarily in the training loss appears. This could have been induced by a number of factors, but the model quickly recovers and continues with the stable convergence phase. The loss depicted in Figure 4.1 shows the best performing model, which is able to successfully reconstruct a spectrogram. A reconstruction by this model of a randomly chosen input spectrogram is shown in Figure 4.2. As seen in Figure 4.2 the model demonstrates a strong ability to reconstruct the input image from only 25% of the original patches. The reconstruction captures the overall spectral structure. Especially, it preserves the key features such as the distinct horizontal frequency bands created by the pump when running in a steady-state. Indicating that the model has learned a meaningful representation of the normal running state.

Taking a look at the error maps from both a normal and an anomalous spectrogram, which can be seen in Figure 4.3. In these error maps it is visible that both a normal spectrogram and an anomalous spectrogram has clear errors, which mostly look like random noise. But for the anomalous case some very significant vertical lines show up, these lines, which also show up in the original images of the anomalous data, show that the model is capable of reconstructing a normal and more general representation of the spectral information in the spectrograms.



(a) Error map generated from normal spectrogram shown in Figure 4.2



(b) Error map generated from reconstruction of an anomalous spectrogram

Figure 4.3: Side by side of error maps generated from reconstructions of, Left: Normal, Right: Anomalous, with clear vertical lines in the error map of the anomalous reconstruction.

As another measure, it is also possible to look at the cumulative sum of errors for each error map to see if the anomalous error maps generate more errors than the error maps of the normal data. The distribution of errors in the two classes can be seen in Figure 4.4.

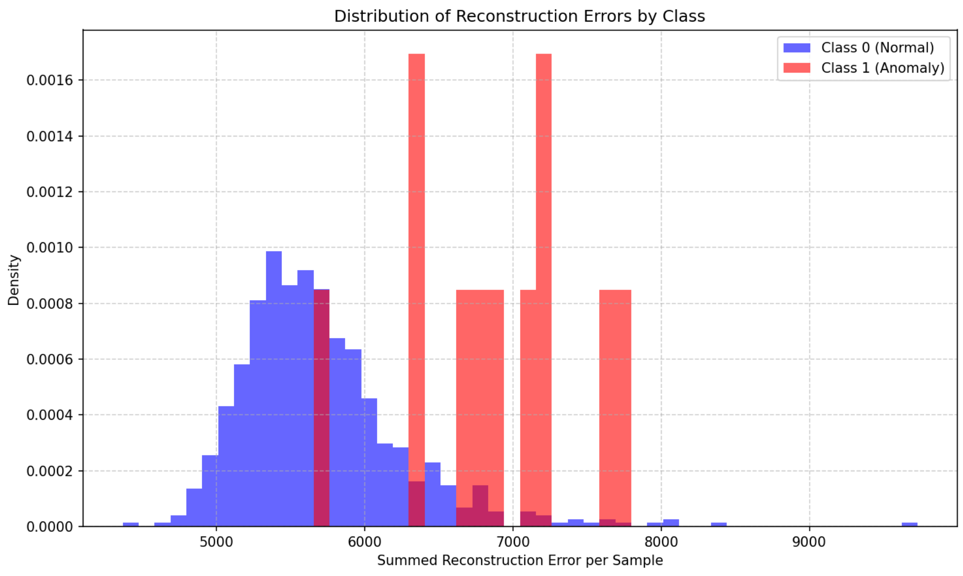
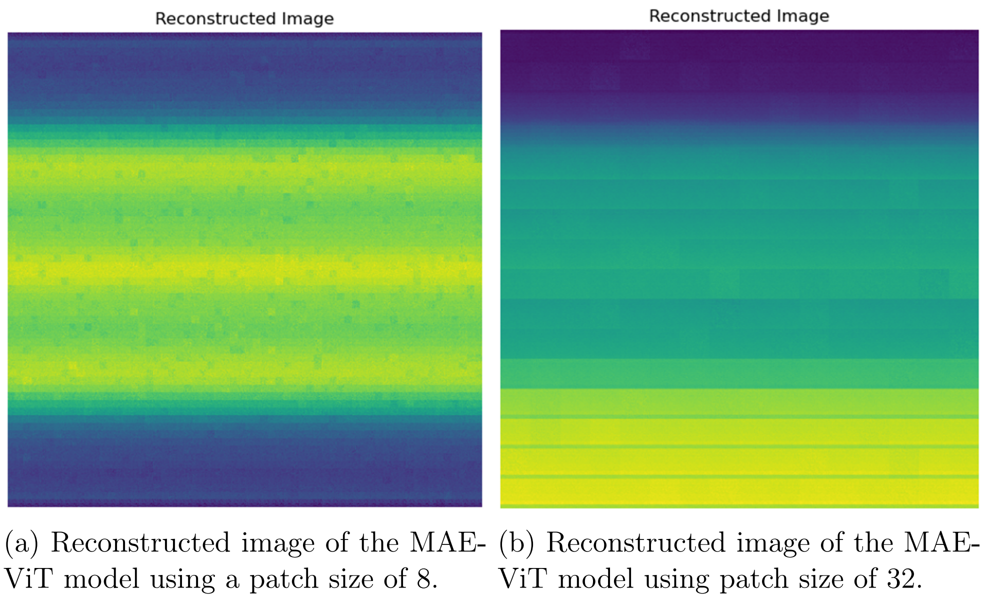


Figure 4.4: Distribution graph showing the total error in each error map for a full validation dataset. Each class graph is mapped such that the densities for each class integrates to 1.

While it is clear to see that the bulk of the two distributions can be mostly separated, but they do still overlap. By calculating the mean errors of the two error distributions, which comes out to $\mu_{normal} = 5697.35$ and $\mu_{anomaly} = 6872.13$. The normal error is clearly lower than the anomaly error. Which gives the fine-tuned downstream task, a good starting point, to separate the two classes, if the fine-tuned model is capable of capturing the information saved in the error maps.

As mentioned in Section 3.3.2 The patch size was tested at different sizes to find the most suitable size for this task. In Figure 4.5.



(a) Reconstructed image of the MAE-ViT model using a patch size of 8.

(b) Reconstructed image of the MAE-ViT model using patch size of 32.

Figure 4.5: Side by side comparison between using a patch size of 8 and 32. The reconstruction of patch size 16 can be seen in Figure 4.2.

The patch sizes of 8 and 32 show two vastly different results, by just visual inspection. The Figure 4.5a shows overgeneralization and suffering from the augmentations of the input images, giving a high validation loss score of 0.011 compared to the low 0.001 score of patch size 16. Figure 4.5b shows a different story, while not suffering from over generalization in the visual inspection, it only captures the very broad frequency bands and then oversimplifies them, with very little fidelity, but surprisingly good validation loss score of 0.001 which is the same for the model using patch size 16.

4.3 Anomaly Detection Results

In this section of the results, the fine-tuned and baseline classifier models will be compared by presenting each of the model results against each other. While a conclusion to these results will be formulated in Chapter 6, it will only be numerically and semantically compared in this chapter.

4.3.1 Learning Dynamics

Starting with the fine-tuned model, it was trained for 200 epochs with a learning rate of 5×10^{-5} and with a batch size of 32, as listed in Table 3.2. It was trained as a downstream task of the MAE-ViT model described in Section 4.2, with all model weights for the MAE-ViT model being frozen in order to only train the fine-tuned model.

The loss curve of the fine-tuned model can be seen in Figure 4.6. In this figure, even though the loss both for the training and validation appears to be jittery, with a seemingly unstable nature. When zooming out across the whole training, a clear downwards trend is visible. Ending with a validation loss of 0.2376 and a training loss of 0.3867.

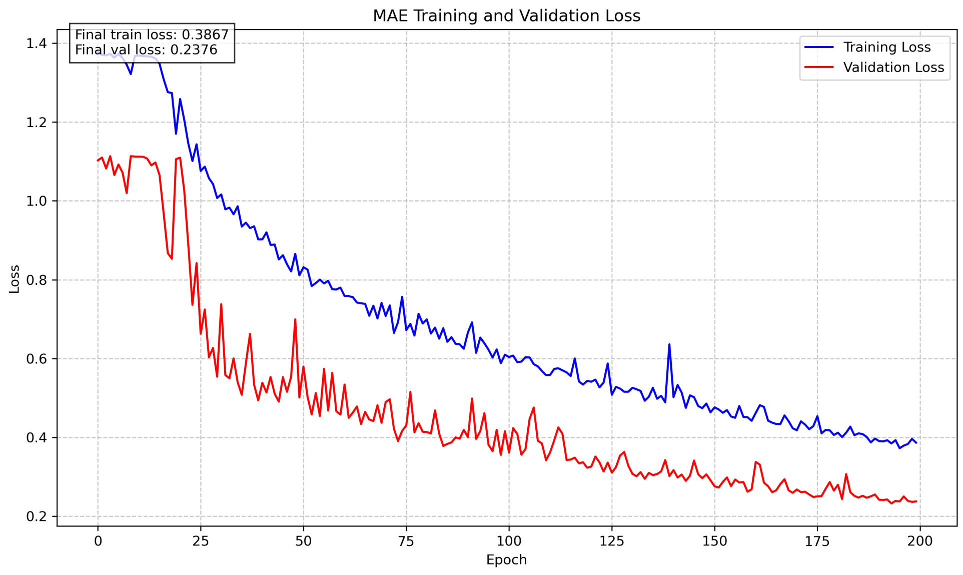


Figure 4.6: Training and validation loss, for the best performing fine-tuned model.

Furthermore, when inspecting the loss curve of the fine-tuned model in Figure 4.6. The loss is seemingly not converging, or rather, it still had a downward trajectory, but a rerun of the model with the same hyperparameters for 300 epochs, seen in Figure 4.7, showed no improvement and only a more unstable validation loss, the rerun ended around the same loss value as the model when run with 200 epochs. Therefore determining that the model performance would not improve with more epochs. The baseline model was set to train for 200 epochs, but due to early stopping the model stopped training at 175 epochs, a learning rate of 1×10^{-5}, a batch size of 32 and with dropout of 0.2. All parameters can be seen in Table 3.3. It was trained from scratch with data as described above in Section 4.1. The loss curve of the best baseline model can be seen in Figure 4.8.

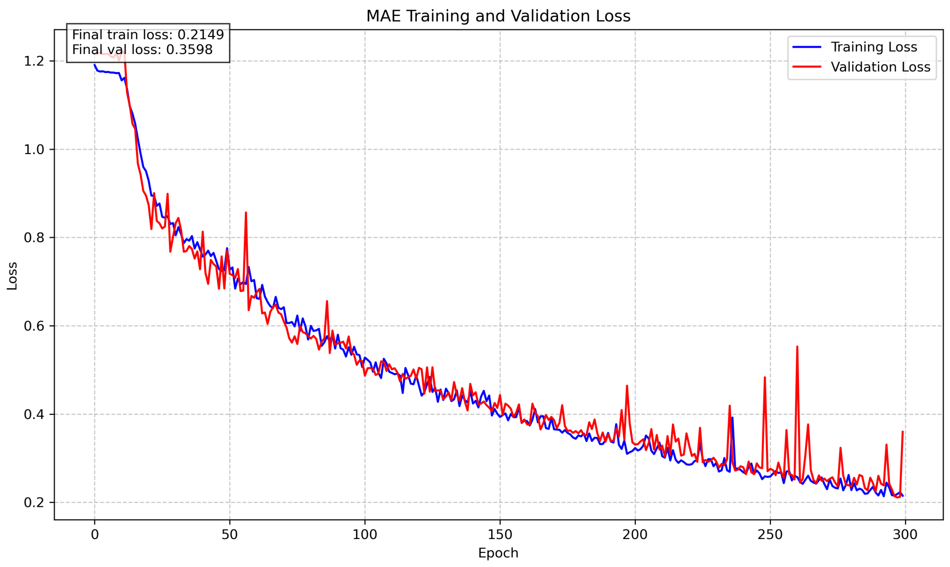


Figure 4.7: Rerun of the fine-tuned model shown in Figure 4.6 with 300 epochs, showing a similar loss curve but more unstable.

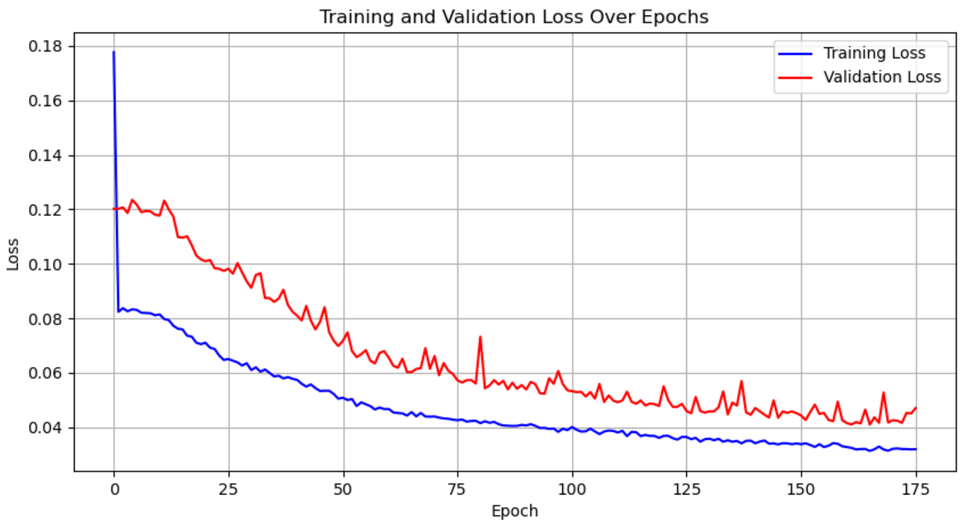


Figure 4.8: Training and Validation loss, for the best performing baseline model.

In the baseline model, a more stable training loss curve can be seen, with the validation loss being more sporadic but still more stable than the fine-tuned model, this baseline model seems to converge towards a stable value. Ending on a training loss of 0.0306 and a validation loss of 0.0443. By taking a closer look at the ending of the graph, the reason for the early stopping can be found, even with a single spike in loss around epoch 168, which seemed quite normal for the model, the continued slow rise after epoch 165 made it stop at epoch 175.

Comparing the losses between the fine-tuned model and the baseline model, the baseline model has a substantially lower loss compared to the fine-tuned model, with around an order of magnitude in difference. This is even though the baseline model has a combination loss of both BCE and MSE. signaling that the baseline model performance, has a chance of being better than the fine-tuned model.

Next metric to look at between the two models is the ROC curve and AUC score. Starting off by looking at the ROC-AUC for the fine-tuned model, seen as the orange line in Figure 4.9.

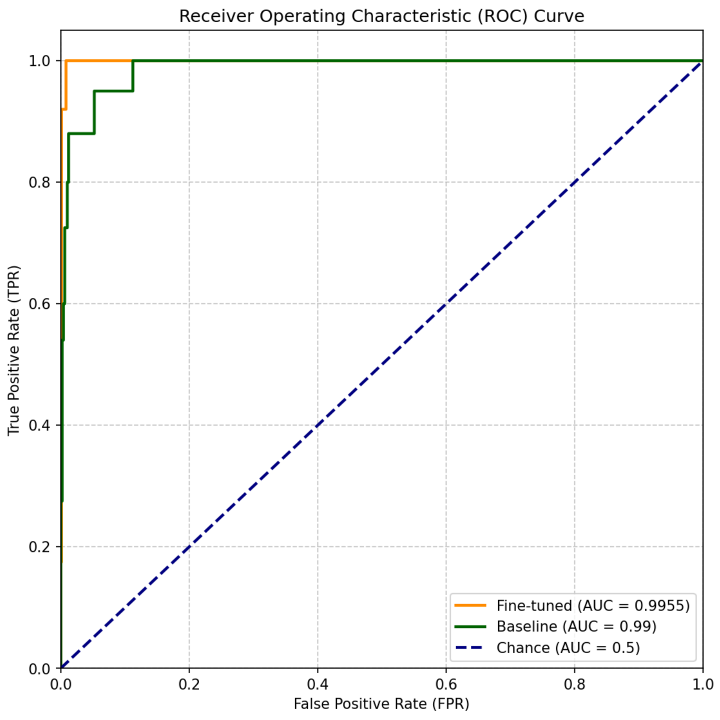
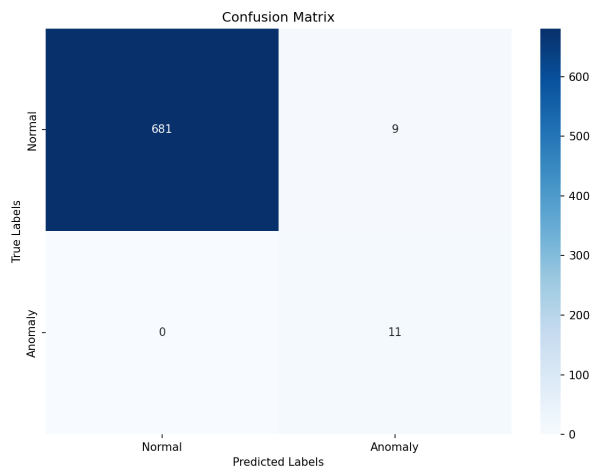


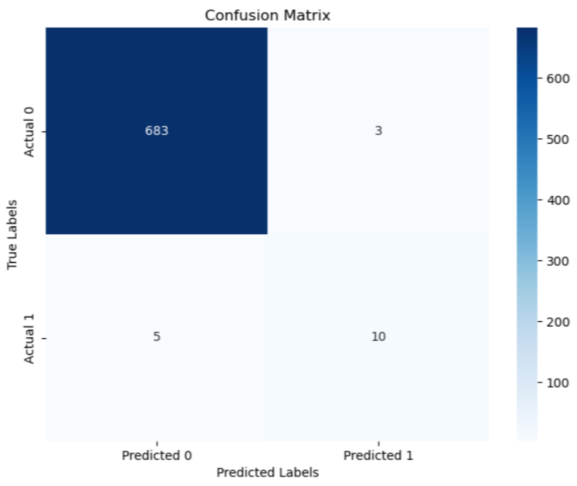
Figure 4.9: ROC curve for both models, with calculated AUC score. The ROC curves are calculated from the same models depicted in Figures 4.6 and 4.8.

Here the performance of the fine-tuned model can be seen as nearly perfect. Across all thresholds, the ROC curve is nearly at a 1.0 TPR. By looking at the calculated AUC score, the performance is summarized by $AUC = 0.996$. This indicates only a few errors when comparing TPR to FPR. Now comparing this to the ROC curve of the baseline model which is shown as the green line in Figure 4.9. Here the ROC curve is also performing very well, with only a slightly worse curve than the fine-tuned model, with a slightly worse TPR. With that said, the summarized AUC score of $AUC = 0.990$ still indicates a very strong performance. So when looking at the ROC-AUC metric, the difference in performance between the two models would indicate two very equal models.

The last of the graphical metrics used to evaluate the two models is the confusion matrix, the matrix for both the fine-tuned and baseline model can be seen in Figure 4.10. Again the performance of the two models again seems very close to equal. The fine-tuned model only misclassifies 9 instances, while the baseline model misclassifies 8 instances.



(a) Confusion matrix for best fine-tuned model.



(b) Confusion matrix for best baseline model.

Figure 4.10: Confusion matrices showing performances for both fine-tuned and baseline model. 0 = Normal, and 1 = Anomaly.

When looking at the individual key values of the confusion matrix, a more nuanced image appears. While the TP and TN predictions are close to equal with 681, 683 and 11 and 10, respectively. A larger difference can be found in the FP and FN predictions. The FP predictions in the fine-tuned model is rather large compared to the TP predictions, respectively 9 against 11, whereas the baseline model only has 3 FP predictions compared to the 10 TP predictions, which when comparing the two models is 3 times as many FP predictions by the fine-tuned model compared to the baseline model. Lastly the fine-tuned model has no FN predictions whereas the baseline model has 5 FN predictions. These differences can be further investigated by looking at the precision and recall values of the two models. The fine-tuned model reaches a precision value of:

$$Precision_{ft} = \frac{11}{11+9} = 0.55 \tag{4.1}$$

Where the precision for the baseline model becomes:

$$Precision_{bl} = \frac{10}{10+3} = 0.77 \tag{4.2}$$

This describes, the fine-tuned model is more likely to predict a normal spectrogram as an anomaly than compared to the baseline model. Whereas looking at the recall the fine-tuned model reaches a value of:

$$Recall_{ft} = \frac{11}{11+0} = 1.0 \tag{4.3}$$

And the baseline model gets a value of:

$$Recall_{bl} = \frac{10}{10+5} = 0.67 \tag{4.4}$$

This shows that the fine-tuned model will never predict an anomaly as a normal spectrogram, whereas the baseline model will miss detecting anomalies 33% of the time. This indicates, by analyzing the values between each model, that they each have their own strong suit. The fine-tuned model captures all anomalies but also catches many false positives. While the baseline model catches most of the anomalies and only a few false positives. And comparing the two models using their respective F1-score, calculated by:

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{4.5}$$

Making the fine-tuned model and baseline model have an F1-score of 0.7097 and 0.7143, respectively. Again, this shows a very even overall performance.

The baseline model was also tested using positive weighting as present in the fine-tuned model. The performance of this model can be seen in Figure 4.11.

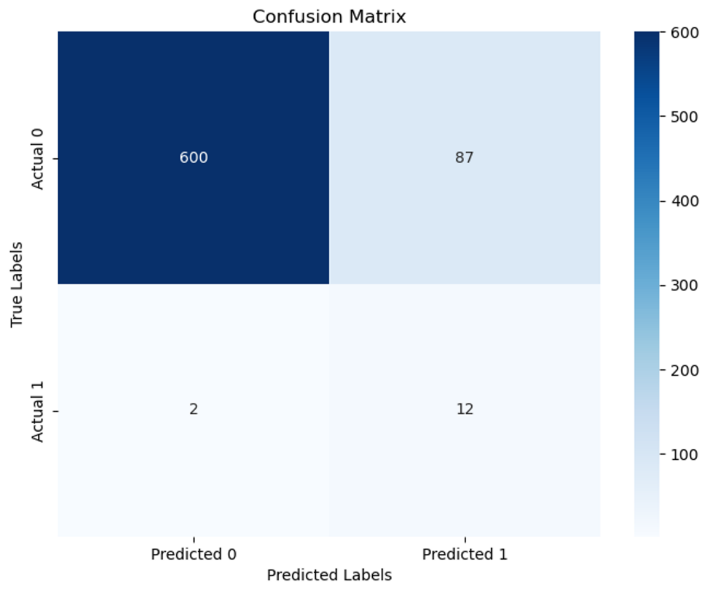
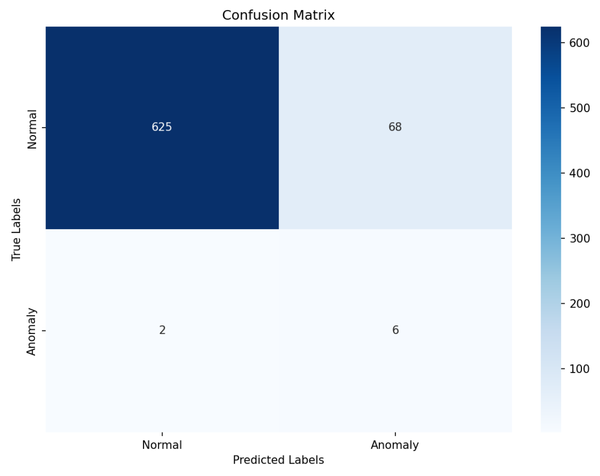


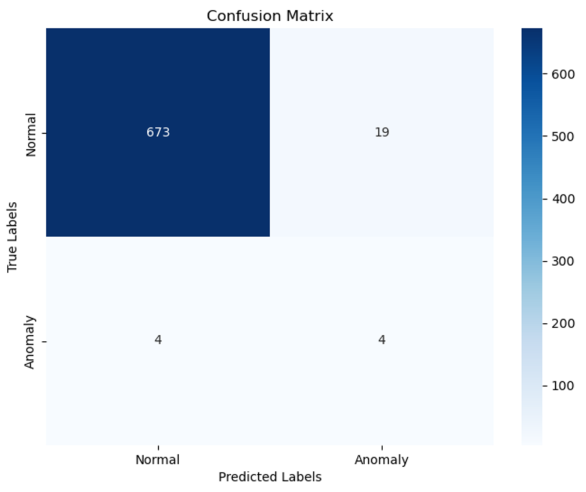
Figure 4.11: Baseline model utilizing positive weighting.

This model performs better in FN predictions with 2 compared to 5 from the chosen baseline model, but because of the positive weighting, it overestimates a great deal of normal spectrograms and gains 87 FP predictions compared to the 3 from the chosen baseline model. Thus showing the positive-weighting for the baseline model underperforming.

As a final test, both the baseline model and the MAE-ViT model have been tested without the CNN classifier attached, which means, they classify each image from the accumulated sum of errors from the error maps, with a threshold decision boundary. These models will be referenced as the simple version of their respective models. This was done to determine the performance of each model, and to see if a complex extra model on top was necessary. The confusion matrix for each of the simple models can be seen in Figure 4.12.



(a) Confusion matrix for the simple MAE-ViT model, using a threshold on the accumulated sum of error from the error maps.



(b) Confusion matrix for the simple baseline model, using a threshold on the accumulated sum of error from the error maps.

Figure 4.12: Confusion matrices showing threshold performances for both the simple MAE-ViT model and the simple baseline model.

In the case of the MAE-ViT model it is clear that it is capable of detecting a good amount of anomalies, and it only has 2 FN predictions just as the baseline model using positive weights. But then, just as the baseline model with positive weights, the model is overcautious and has 68 FP predictions. Whereas, the simple baseline model using a threshold decision boundary has 4 TP prediction, which is an underperformance compared to the other models, but the simple baseline model is not being as overly cautious as the simple MAE-ViT model.

The performance metrics of all the mentioned models in this chapter, can be seen in Table 4.1.

4.4 Final Evaluation

Summarizing the results and the findings to give a final evaluation of the models, and assess if the MAE-ViT model and the fine-tuned model, lived up to, and fulfilled the research questions, which was set out to be tested and investigated.

Image Reconstruction Evaluation

Looking at the loss curve of the MAE-ViT model in Figure 4.1, the model has converged to a stable loss value signifying that it has learned to reconstruct the normal representation of a spectrogram to the best of its abilities with the current setup and hyperparameters. Further support for this result comes from the error map distribution in Figure 4.4, which shows a good separation between the mean sum of errors in a normal reconstruction compared to the reconstruction of the anomalous case. Moreover, by visual inspection of the reconstructed image in Figure 4.2 the model is capable of capturing the general structure of a spectrogram, with the ability to separate the general frequency bands visible in the original spectrogram. Which means that before evaluating the complete system, the SSL algorithm using MAE framework with ViT, is capable of reconstructing a normal representation of a running pump spectrogram to a sufficient enough degree such that it can be used as a pre-trained model, for downstream tasks.

Image Classification Evaluation

Summarizing the comparison between the fine-tuned model and baseline model. The baseline model starts off by showing substantially better performance in the validation loss, when comparing Figure 4.8 and 4.6. But looking at more classification specific metrics, the fine-tuned model outperforms the baseline model by a slim margin in the ROC-AUC evaluation in Figure 4.9. Here the fine-tuned model scores $AUC = 0.996$ against the baseline models $AUC = 0.990$. This indicates a slightly better classification performance by the fine-tuned model. Lastly, looking at the confusion matrices for both models in Figure 4.10, the fine-tuned model outperforms the baseline model, with the fact that it correctly classified all anomalies in its validation dataset, compared to the baseline model that only classified anomalies correct 66% of the time. In a company setting such as this case with Grundfos it is clear that not being able to correctly identify all anomalies can lead to faulty systems reaching production, which is an unwanted scenario.

For a final comparison, all the tested models have been collected and summarized in Table 4.1. This table gives a short overview of all the different models, and their respective performances.

Name	**Dataset**	**Decision method**	**Val Loss**	**AUC**	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1-Score**
Fine-tuned	Full	CNN	0.238	0.996	11	681	9	0	0.55	1	0.71
Baseline	Full	CNN	0.044	0.990	10	683	3	5	0.77	0.67	0.71
Baseline Pos Weight	Full	CNN	0.437	0.960	12	600	87	2	0.12	0.86	0.21
Baseline Threshold	Full	Threshold	0.044	0.690	4	673	19	4	0.17	0.50	0.25
MAE-ViT	Normal	Threshold	0.001	0.820	6	625	68	2	0.08	0.75	0.14

Table 4.1: Comparison table with evaluation metrics for all tested models. Normal dataset is referring to it only being trained on the 7095 normal datapoints.

This is leading to the question: Can a CNN used as a downstream task of a pre-trained SSL model correctly classify errors in pump data? And looking at the results, it correctly classified all anomalies used in this dataset. Furthermore, can it outperform a Convolutional Autoencoder? Given the results showing the Convolutional Autoencoder only correctly classified 66% of the anomaly cases, shows that it did in fact outperform the Convolutional Autoencoder used as a baseline model for this thesis.

5 | Discussion

This discussion chapter aims to further elaborate on topics and decisions made throughout this thesis. The topics chosen were deemed the most important for this particular thesis.

5.1 Dataset Uncertainties

While collecting the data for this thesis, the first major hurdle appeared, the data had no clear annotations if the data included an error. This missing annotation meant that all collected data had to be searched through. By utilizing keyword searching the comments on each collected dataset, all datasets annotated with cavitation errors in the data comments were collected. Although all 95 known cavitation errors were collected, the data classified as normal for this thesis, should contain no such errors. However, it cannot be certain that no other anomalies are featured in the normal dataset. These possible unknown anomalies could have caused the MAE-ViT model to perform worse in the reconstruction loss score, due to unknown errors affecting the reconstruction of a normal spectrogram, creating unwanted artifacts in the error map. These artifacts could also impact the performance of the downstream task. Which could explain some of the model classification uncertainty.

5.2 Spectrogram Image Scaling

When creating the spectrograms from the now classified audio signals, it was a conscious decision to keep the spectrogram images at the size (496×496) instead of down-scaling them to the more commonly used size of (224×224), popularized by the Deep CNN, AlexNet [3]. While this decision evidently would increase the computational time by increasing the model parameters. It was chosen to retain a high fidelity because cavitation anomalies are visible as transients, with a broad frequency band in the spectrograms, which should be more easily identifiable with a larger image. Though an experiment to verify the use of a larger image to have a positive effect on performance and outweighing the optimization penalty was not conducted. Therefore, it cannot be for certain that this was the most optimal approach for this thesis.

To add to the image size and fidelity discussion, when creating the spectrograms a window function, with a small window size was used, this was to further emphasize the importance of the time components. Making it so that the spectrogram is able to capture multiple transients occurring in very close proximity. The ability to capture more transients, will for the cavitation errors increase the sum of errors in the error map, and in turn help the down stream task to identify the cavitation errors better.

5.3 Baseline Positive Weighting

In Section 3.4.3 it was presented that the baseline model had a significantly better loss score, than that of the fine-tuned model, by an order of magnitude. however, further analysis showed that this lower loss did not directly translate to better performance. While the two models had comparable loss functions, with the baseline model utilizing both BCE loss and MSE loss. The addition of MSE loss only had a small impact on the total loss for the baseline model, because it was an order of magnitude lower than that of the BCE loss. Which then shifts focus to the addition of positive weighting for the fine-tuned model. This positive weighting ensured that the positive samples or anomalies had a larger impact on the loss function because of the very few positive samples. This positive weight therefore also scaled the loss values from the anomalies, which in turn would also increase the value of the loss used to train the fine-tuned model. When adding positive weighting to the baseline model, the validation loss increased to the same level as that of the fine-tuned model, but as presented in Section 4.3 this addition of positive weighting was omitted from the baseline model, because the model only saw a decrease in performance when introduced.

5.4 Decoder Utilization

A design choice that differs from the standard utilization of autoencoders is the use of the decoded reconstructions as a part of the downstream task. Whereas, a standard use would be to utilize the encoded latent space as the input for the downstream task, as done in [13] and [6]. This way the decoder can be omitted and the input vector for the downstream task is then a set of compact features from the latent space, able to represent the full input. But for the anomaly detection objective of this thesis, this representation was not sufficient. Because when dealing with anomaly detection and especially with sparse anomalies, it is very hard for a SSL model to correctly create a latent representation of anomalies it rarely sees. It was therefore decided that the MAE-ViT model would be trained to create a normal representation of the input image, and then use the difference between the input and reconstruction, to capture information about anomalies. Error maps showed to be rather effective for the fine-tuned model in order to separate anomalies from normal data, and outperformed the models while only using thresholding, which is why no other alternative was tested.

5.5 Alternative Baseline Training Approach

When creating the baseline model, its sole purpose was to use it as a classification model to rival the downstream fine-tuned model. Therefore it was built and trained as a classifier. Meaning, the baseline model was trained on the complete dataset with both normal and anomalous data samples to try and distinguish between the two classes. Even though the baseline model is an autoencoder, the reconstruction part of the model was only meant as a loss parameter and not a comparison metric.

Another approach could have been to start by training the baseline model just as the MAE-ViT model on only normal data, and then have compared these two models and their respective reconstruction capabilities, and after, just like the fine-tuned model, freeze the decoder and then trained the downstream CNN classifier of the baseline model as with the fine-tuned model. This way the baseline model was better suited at directly comparing between both the MAE-ViT model and the fine-tuned model. The comparison made in this thesis is still sufficient for the binary case and when anomalous data is present. However, if the model is to be extended to handle data from a source where no anomalous data is present, the discussed approach above would be a better fit.

5.6 Importance of Recall

As mentioned in Section 3.4.3, a larger emphasis was put on recall when evaluating performance of the baseline and fine-tuned model. While other factors do certainly also have an impact. For this very case-specific thesis of helping a testing facility to detect anomalies in pump data, a recall value of 1 should be the most important evaluation factor. This is in line with the evaluation from the engineers and technicians at S&V Lab, stating that not detecting an anomaly for a testing facility could mean finalizing a testing report and approving the pump even though it has a clear performance–reducing error, that should be addressed or a reduced lifespan. Such errors could be costly if the pump malfunctions earlier than the expected lifespan, which would generate costly reparations or complete replacement of the pump after installation. This would drive up the spending and reduce the profit.

The reduced testing efficiency in the model being overly cautious would still be preferred compared to missing anomalies and labeling them as normal data. This is why for this very case-specific task, a higher recall value would for a reasonable margin always be preferred, when being implemented into a work environment. The reasonable margin is not set in stone, and could always be argued, which is the reason why a model as the baseline with positive weighting, is still not preferred to the chosen baseline model, because even with a better FN rate with 2 compared to 5, the FP rate of 87 compared to 3 is a clear degradation of model performance, that cannot be excused by the higher recall.

6 | Conclusion

This chapter aims to bring the thesis to a conclusion by addressing the research questions outlined in Section 2.2 and shine a light on possible future works.

The first research question was stated as follows: *"Can a Self-Supervised Machine Learning algorithm, built on a Masked Autoencoder framework, utilizing Vision Transformers, reconstruct a normal representation of a spectrogram of a running pump?"*

As this is a rather subjective research question with no ground truth. Answering it in isolation from the rest of the thesis, means that the reasoning when concluding this question will solely be on the performance of the SSL MAE-ViT model with no downstream task attached. With the validation curve shown in Figure 4.1, which shows a rather stable learning curve with two major drops in loss, indicating the model learned defining features of the data at these points. The ending validation loss of 0.001 shows that some sort of resemblance between the input image and reconstruction has been reached.

This is further supported, when analyzing the reconstructed images in either Figure 3.4 or 4.2. These figures show the models ability to reconstruct frequency bands to a certain fidelity, and even though it is not perfect, it has the correct structure, which is the main capability wanted, when reconstructing a normal representation of the signal. By looking at Figure 4.4, a better insight into the model's ability to reconstruct the normal representation emerges. With a mostly clear distinction between the sum of errors in a normal spectrogram and a anomalous spectrogram. The model clearly omits most of the behaviors from the anomalous data when reconstructing the image, which gives a higher sum of individual errors in that type of data. This now brings the conclusion regarding the first research question. Yes, the SSL MAE-ViT model is capable of reconstructing a normal representation of a spectrogram from a running pump, to the extent that it is needed for this thesis.

The second research question was stated as follows: *"Can a Convolutional Neural Network correctly classify errors in pump data, when used as a downstream task of a pre-trained Self-Supervised Learning model?"*

This research question is more easily answered from the results than the former question. The fine-tuned model, which is a 3-layer CNN model used as a downstream task built on top of the MAE-ViT model. Looking at the confusion matrix in Figure 4.10a a clear image emerges. With no FN cases, the model is capable of detecting all anomalies with a recall value of 1, and an AUC score of 0.996, the only downside of the model is the rather low precision score of 0.55. But for the specific use-case of this thesis, the Precision value is not weighted as highly as the Recall. Therefore, the conclusion to the second research question is: Yes, the model can correctly classify all errors present in the dataset provided for this thesis.

The third and last research question was stated as follows: *"Can the Convolutional Neural Network outperform a Convolutional Autoencoder trained on the same data?"*

This question is closely tied to the second question, but now rather than just completing the task it has to outperform another model. The other model, a Convolutional autoencoder, has shown great performance at similar tasks to this one. While both models perform quite well in this given task, the metrics also show a close comparison between them. Starting with the loss curves, the baseline model shows a lower loss than the fine-tuned model, which was a great performance, but as discussed in Chapter 5, due to the implemented positive weighting on the fine-tuned model, which punishes the model harder, when it misclassifies anomalous data, this increases the loss. Therefore, this direct comparison between the two models, might not be fair.

The next metric the ROC curve and AUC-score. The figures referenced here can be seen in Figure 4.9. These values should be more directly comparable, and both models have very good performances here. With the fine-tuned model performing slightly better than the baseline model, with an AUC score of 0.996 against the 0.990 score from the baseline model. Lastly and the deciding factor for the conclusion on which of the models are better. The confusion matrices seen in Figure 4.10 shine light on this deciding factor. As discussed in Chapter 5 and mentioned when concluding the second research question, the fine-tuned model performs to a recall value of 1, whereas the baseline model only achieved a recall value of 0.67 in its overall best performance. With most of the other performance metrics gained from the confusion matrix being close to equal, the conclusion to the research question becomes: Yes, the fine-tuned CNN model, is able to outperform the baseline Convolutional autoencoder model, with both models being trained from scratch on the same data.

6.1 Future work

This future work section aims to identify and suggest further improvements or research into this thesis that may have been outside the scope of this thesis, or is realizations made during the work on this thesis, but could not be performed within the time frame.

As stated in Section 3.2, only a single-channel audio recording was used for creating spectrograms, more precisely only the right channel. This was decided together with the team at S&V Lab, because the signals were close to identical and the right channel is placed on the same side as the pump housing, where the audible error should be most prominent. But as it was not tested, it is not possible to say if any information got lost from missing the left channel. Therefore, a possible test case would be to include both left and right audio channel when creating spectrograms, to see if this further emphasizes both the normal behavior and the abnormal behavior. It would increase the amount of information gathered for the project, but as both channels are combined to a single spectrogram, this should not change the data needed to be handled by the models therefore not sacrificing training and validation calculation time. This is a small but interesting change to the information in which the models determine their decisions from, that could have a larger impact than firstly anticipated.

As described in Section 3.2 the spectrogram images were created at size (496×496) with the reasoning of retaining as much information as possible. But as later discussed in Chapter 5, it is possible to downscale the image to a smaller size to reduce the computation time for training and validation. It would be interesting to see if reducing the image size would reduce the model performance or if it retained the same information and performance and did so with faster training and validation. This reduction in image size would also impact the patch size, and the relative patch size to image size, which should also be a consideration for when reducing the image size, that the patch size should also be changed to a more suitable size.

The models proposed in this thesis are all binary anomaly detection models, detecting either an anomaly or not. However, for a company as Grundfos, they may want to expand on this idea. Instead of just detecting an anomaly, they might want a model that can identify what type of anomaly it detects, increasing the complexity to a multiclass classification problem. As the MAE-ViT model is trained to reconstruct the normal representation of a spectrogram, no matter the anomaly, this model should not need any changes to increase the amount of different anomalies. The downstream fine-tuned model is where most of the changes would happen. A few different scenarios should be considered. First, if the anomalies that should be distinguished between have vastly different spectrogram features, then the fine-tuned model should just be trained with the corresponding output nodes in order to gain an understanding of the new anomalies. But if the spectrograms do not have easily distinguishable features. This would likely cause a complete reconstruction of the fine-tuned model. This could include increasing the depth of the model in order to gain more complex features from the error maps, or perhaps change the approach, and use something other than a simple error map, or change the model to something other than a CNN.

Bibliography

[1] J. Thilagavathi. “Deep Learning for Smart Manufacturing: Methods and Applications”. In: *International Journal of Intelligent Systems and Applications in Engineering* 12.22s (2024), 891 –. URL: `https://ijisae.org/index.php/IJISAE/article/view/6572`.

[2] Forbes. *How Non-Tech Companies Are Leveraging Machine Learning.* Last accessed: 2025-03-14. 2023. URL: `https://www.forbes.com/sites/forbesbusinesscouncil/2023/03/14/how-non-tech-companies-are-leveraging-machine-learning/`.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Commun. ACM* 60.6 (May 2017), 84–90. ISSN: 0001-0782. DOI: `10.1145/3065386`. URL: `https://doi.org/10.1145/3065386`.

[4] Kaiming He et al. *Deep Residual Learning for Image Recognition.* 2015. arXiv: `1512.03385 [cs.CV]`. URL: `https://arxiv.org/abs/1512.03385`.

[5] Ashish Vaswani et al. *Attention Is All You Need.* 2023. arXiv: `1706.03762 [cs.CL]`. URL: `https://arxiv.org/abs/1706.03762`.

[6] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* 2021. arXiv: `2010.11929 [cs.CV]`. URL: `https://arxiv.org/abs/2010.11929`.

[7] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations.* 2020. arXiv: `2002.05709 [cs.LG]`. URL: `https://arxiv.org/abs/2002.05709`.

[8] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning.* 2020. arXiv: `1911.05722 [cs.CV]`. URL: `https://arxiv.org/abs/1911.05722`.

[9] Robert Müller et al. “Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning”. In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART.* INSTICC. SciTePress, 2021, pp. 49–56. ISBN: 978-989-758-484-8. DOI: `10.5220/0010185800490056`.

[10] Sanyuan Chen et al. *BEATs: Audio Pre-Training with Acoustic Tokenizers.* 2022. arXiv: `2212.09058 [eess.AS]`. URL: `https://arxiv.org/abs/2212.09058`.

[11] Phurich Saengthong and Takahiro Shinozaki. *Deep Generic Representations for Domain-Generalized Anomalous Sound Detection*. 2024. arXiv: `2409.05035 [cs.SD]`. URL: `https://arxiv.org/abs/2409.05035`.

[12] Marco Tagliasacchi et al. "Pre-Training Audio Representations With Self-Supervision". In: *IEEE Signal Processing Letters* 27 (2020), pp. 600–604. DOI: `10.1109/LSP.2020.2985586`.

[13] Kaiming He et al. *Masked Autoencoders Are Scalable Vision Learners*. 2021. arXiv: `2111.06377 [cs.CV]`. URL: `https://arxiv.org/abs/2111.06377`.

[14] Erik Marchi et al. "A Novel Approach for Automatic Acoustic Novelty Detection Using a Denoising Autoencoder with Bidirectional LSTM Neural Networks". In: *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing* (Apr. 2015).

[15] Manassés Ribeiro, André Eugênio Lazzaretti, and Heitor Silvério Lopes. "A study of deep convolutional auto-encoders for anomaly detection in videos". In: *Pattern Recognition Letters* 105 (2018). Machine Learning and Applications in Artificial Intelligence, pp. 13–22. ISSN: 0167-8655. DOI: `https://doi.org/10.1016/j.patrec.2017.07.016`. URL: `https://www.sciencedirect.com/science/article/pii/S0167865517302489`.

[16] Gopikrishna Pavuluri and Gayathri Annem. *A Deep Learning Approach to Video Anomaly Detection using Convolutional Autoencoders*. 2023. arXiv: `2311.04351 [cs.CV]`. URL: `https://arxiv.org/abs/2311.04351`.

[17] Francisco J. Martinez-Murcia et al. "Studying the Manifold Structure of Alzheimer's Disease: A Deep Learning Approach Using Convolutional Autoencoders". In: *IEEE Journal of Biomedical and Health Informatics* 24.1 (2020), pp. 17–26. DOI: `10.1109/JBHI.2019.2914970`.

[18] Brüel & Kjær. *Sound Quality Head and Torso Simulator Type 4100 & 4100 D*. `https://media.hbkworld.com/m/0d12f49787dbd303/original/Sound-Quality-Head-and-Torso-Simulator-Type-4100-4100-D.pdf`. Last accessed: 2025-03-17. 2013.

[19] Brüel & Kjær. *PRODUCT DATA - $\frac{1}{2}$" Prepolarized Free-field Microphone - Type 4189*. `https://media.hbkworld.com/m/2019bebd14274daf/original/Datasheet-Prepolarized-Free-field-Microphone-Type-4189-BP2210.pdf`. Last accessed: 2025-03-17. 2008.

[20] Brüel & Kjær. *LAN-XI Data Acquisition Hardware*. `https://www.bksv.com/media/doc/bp2215.pdf`. Last accessed: 2025-05-01. 2024.

[21] Brüel & Kjær. *Software for PULSE LabShop*. `https://www.bksv.com/media/doc/bu0229.pdf`. Last accessed: 2025-05-01. 2018.

[22] International Organization for Standardization. *ISO 3745:2023 Acoustics Determination of sound power levels and sound energy levels of noise sources using sound pressure Precision methods for anechoic rooms and hemi-anechoic rooms*. Standard. Paid standard, Last accessed: 2025-04-29. International Organization for Standardization, 2023.

[23] Open-source. *SciPy (Version 1.15.2)*. `https://scipy.org/`. Python library. 2025.

[24] Zheng-Hua Tan. "Lecture 9: Deep Learning". Lecture slides from Machine Learning course E23. 2023.

[25] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: `10.1162/neco.1989.1.4.541`.

[26] Li Deng and Dong Yu. *Deep Learning: Methods and Applications*. Now, Foundations and trends, 2014. DOI: `10.1561/2000000039`.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[28] PyTorch Contributors. *Conv2d*. `https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html`. Last accessed: 2025-05-06. 2025.

[29] PyTorch Contributors. *ReLU*. `https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html`. Last accessed: 2025-05-06. 2025.

[30] PyTorch Contributors. *AvgPool2d*. `https://pytorch.org/docs/stable/generated/torch.nn.AvgPool2d.html`. Last accessed: 2025-05-06. 2025.

[31] PyTorch Contributors. *BCEWithLogitsLoss*. `https://docs.pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html`. Last accessed: 2025-05-08. 2025.

[32] David Hand and Peter Christen. "A note on using the F-measure for evaluating record linkage algorithms". In: *Statistics and Computing* 28.3 (2017). http://spiral.imperial.ac.uk/b
d-16-00349-final.pdf, pp. 539–547. DOI: `10.1007/s11222-017-9746-6`. URL: `https://app.dimensions.ai/details/publication/pub.1084928040`.

[33] Python Software Foundation. *Python*. `https://www.python.org/`. Coding language used. 2024.

[34] Linux foundation. *torch (Version 2.5.1)*. `https://pytorch.org/`. Used for setting up Machine Learning algorithms. 2024.

[35] William Falcon. *Lightning (Version 2.5.1)*. `https://lightning.ai/docs/pytorch/stable/`. Used for setting up Machine Learning algorithms. 2024.

[36] Inria. *Scikit-learn (Version 1.6.1)*. `https://scikit-learn.org/stable/index.html`. Python library. 2025.

[37] Open-source. *Numpy (Version 2.0.1)*. `https://numpy.org/`. Python library. 2024.

[38] Open-source. *Matplotlib (Version 3.10.0)*. `https://matplotlib.org/stable/index.html`. Python library. 2024.

[39] Michael Waskom. *Seaborn (Version 0.13.2)*. `https://seaborn.pydata.org/`. Python library. 2024.

I | Python code

All code used for this thesis can be found on GitHub following the link: `https://github.com/Dronge1996/Master_Thesis.git`