Pseudo-anomaly generation for improving the unsupervised anomaly detection task

Implementation of a generative nerual network

Andreas Bach Daasbjerg

Vision Graphics and Interactive Systems, VGIS10-1044, 2023-06

Master's Project

AALBORG UNIVERSITY

STUDENT REPORT

Department of Electronic Systems - Electronics and IT
Aalborg University
Fredrik Bajers Vej 7
https://www.es.aau.dk/

Title:
Pseudo-anomaly generation for improving the unsupervised anomaly detection task

Theme:
Generative modelling

Project Period:
Spring Semester 2023

Project Group:
1044

Participant(s):
Andreas Bach Daasbjerg

Supervisor(s):
Neelu Madan
Kamal Nasrollahi

Copies: 1

Page Numbers: 82

Date of Completion:
June 1, 2023

Abstract:

Anomaly detection can be seen as a one-class classification problem, due to the rare occurrence of anomalous frames. Introducing pseudo-anomalies to the training set can potentially improve the performance of the anomaly detection model. This project introduces a pipeline for improving the unsupervised anomaly detection task, by teaching a generator to generate pseudo-anomalies. The pseudo-anomalies are generated by increasing a loss component in the overall loss function of the generator. Two loss components were tested, kullback-leibler divergence and flow loss. The pseudo-anomalies are used to train a classifier to classify normal and abnormal frames. Upon evaluation of the model, an AUC score is calculated using a combination of the classification score and the psnr score. The pipeline achieved an AUC-score of 72.42% evaluated on the CUHK Avenue dataset. While not achieving state-of-the-art results, the pipeline shows potential for improving the performance of anomaly detection tasks. Future work could include adding a second generator branch to generate normal frames. Another approach is to evaluate the performance of the pipeline on a different dataset, such as the ShanghaiTech dataset.

Contents

Preface **viii**

1 Introduction **1**
1.1 Initial problem statement 2

2 Literature Review **3**
2.1 Artificial Neural Networks 3
2.1.1 Neurons 4
2.2 Neural Networks 6
2.2.1 Common Neural Networks 7
2.3 Generative Neural Networks 8
2.3.1 Generative Adversarial Networks: 9
Application of GANs 9
2.3.2 Diffusion Probabilistic Models (DPM): 10
Remarks on diffusion model usage 11
2.4 Final problem statement 11

3 Proposed System **12**
3.1 Architecture overview 12
3.1.1 Generator architecture 12
3.1.2 Anomaly feature extraction 12
3.1.3 Normal feature extraction 13
3.1.4 Classification 14
Training the classifier network 14
AUC-score modification 14
3.2 Baseline Methods 15
3.2.1 Baseline 1: Future Frame 15
3.2.2 Baseline 2: OGNet 16

4 Experimental Results **18**
4.1 Evaluation metrics 18

4.1.1 Performance measurement . . . 18
4.1.2 Dataset . . . 18
4.1.3 Training the pipeline . . . 19
4.1.4 Sofware . . . 19
4.1.5 Hardware . . . 19
4.2 Diffusion model experiment . . . 19
4.3 Baseline experiments . . . 20
4.3.1 Preliminary auto-encoder result . . . 20
4.3.2 Future Frame Prediction result . . . 20
Prediction of future frames . . . 21
4.3.3 Preliminary OGNet result . . . 21
4.4 Experiments with proposed model . . . 23
4.4.1 GAN result . . . 23
4.4.2 Using KL divergence and perceptual loss . . . 25
Experiment 1: 0.1 KL divergence loss . . . 25
Experiment 2: 0.0001 KL divergence loss . . . 26
Experiment 3: 0.0001 KL divergence loss with perceptual loss 26
Experiment 4: 0.0001 KL divergence loss with perceptual loss and regularisation . . . 27
4.4.3 Inverting flow loss . . . 29
Experiment 1: 0.5 flow loss . . . 29
Experiment 2: 0.6 flow loss . . . 31
Experiment 3: 0.7 flow loss . . . 32
Pseudo-experiment: Inverting inputs of flow loss calculation 34
4.4.4 Main takeaways . . . 35
Kl-loss . . . 35
Flow loss . . . 35
4.5 Evaluating pipeline . . . 35
4.5.1 Learning rate test . . . 36
4.5.2 Quantitative result . . . 36
4.5.3 Qualitative result . . . 37
Detections in videos with both normal and abnormal frames 37
Detections on videos with only abnormal frames . . . 41
4.5.4 Main takeaway from the qualitative results . . . 45
4.6 Discussion of results . . . 45
4.6.1 Vanishing foreground objects in GAN architecture . . . 45
4.6.2 Problems with kl-loss application . . . 45
Results from other tests . . . 46
4.6.3 Inversion the flow loss . . . 47
Larger inversion factors . . . 47
Issue when inverting the flow loss . . . 48

4.6.4 Classifier . . . 48
4.6.5 Results of pipeline evaluation . . . 49
Quantitative results . . . 49
Qualitative results . . . 50
Improvement to the proposed system . . . 51
4.7 Future work . . . 52

5 Conclusion **53**

Bibliography **56**

A Resources **63**
A.1 Github repository . . . 63

B Prior report **64**
B.1 Prior work . . . 66
B.1.1 Image synthesis for action localisation . . . 66
B.1.2 Image Synthesis: RGB to Thermal Domain . . . 66

C Diffusion Models **70**
C.1 Recent developments in image synthesis . . . 70
C.1.1 Diffusion model framework . . . 70
C.1.2 Improved Denoising Diffusion Probabilistic Models . . . 71
GAN Comparison . . . 72
Model training . . . 73
C.1.3 UNIT-DDPM: Unpaired Image Translation with Denoising Diffusion Probabilistic Models . . . 73
UNIT-DDPM evaluation . . . 74
C.1.4 Remarks on diffusion model usage . . . 75

D Qualitative Results **77**
D.1 Qualitative results of the proposed system . . . 77
D.1.1 Video 3 . . . 77
D.1.2 Video 4 . . . 78
D.1.3 Video 6 . . . 79
D.1.4 Video 17 . . . 80
D.1.5 Video 18 . . . 81
D.1.6 Video 21 . . . 82

Preface

This master thesis was written during the tenth semester of Vision Graphics and Interactive Systems (VGIS) at Aalborg University within the Department of Electronic Systems. The focus of this report is to research into generating pseudo-anomalies for improving the unsupervised anomaly detection task. The writing of this report spanned the entierty of the 10th semester from February to the 2nd of June 2023.

Citation Usage and Style:

For citations to the bibliography the following applies: when a citation is made before a period the citation applies to the previous sentence(s), a citation made after a period applies to the entire paragraph. In the bibliography the reference are numbered following the IEEE referencing format. The entries within the bibliography contain, at minimum, the following: author(s), title, year, if the reference was found online the URL is provided, as well as the retrieval date.

Aalborg University, June 1, 2023

Andreas Bach Daasbjerg
<adaasb18@student.aau.dk>

1 Introduction

The anomaly detection task is a problem of one-class classification, due to the rare occurrence of anomalous scenes. Anomaly detection is also often viewed as a novelty detection problem, in which a model is trained based on known class data to detect outliers as abnormal [70]. Moreover, with an increase in the popularity of deep learning, instead of hand-picking the features for anomaly detection tasks, researchers propose to use pre-trained convolutional network-based features to train once-class classifiers [70]. Anomaly detection has a wide range of applications in many different fields and is a cross-industry concern [16]. In the financial sector, uses of anomaly detection include asset pricing and forensic accounting (the process of checking if fraudulent actions have occurred) [16]. In the field of medicine, anomaly detection sees its use in the detection of diseases and other medical conditions. Such a task could be determining whether a person is healthy or unhealthy based on their measurements versus the norm [16]. In video surveillance, anomaly detection refers to identifying events that are not conforming to expected behaviour. These unexpected behaviours or anomalies are used to determine suspicious activity and potentially prevent crime [39]. Data is gathered from CCTV cameras or other surveillance sensors. The data represents the behaviour of the surveillance targets in which some behaviour is assumed to be outside the norm [65]. A feature extraction process is applied to the raw data. These features describe the behaviour of the surveillance targets. Once these features are applied to a learning model, the state of the observed target can be determined and labelled as normal or abnormal [65]. The raw data, along with the labels, can then be combined into a dataset that can be used for anomaly detection. Such datasets include the CUHK Avenue dataset [40], the USCD Pedestrian dataset [44] and the ShanghaiTech and ShanghaiTech Campus datasets [76, 39].

These datasets contain both normal and abnormal behaviour of both pedestrians and the surroundings of the area in which the data was gathered. Figure 1.1 shows two examples of abnormal behaviour from the CUHK Avenue dataset. Figure 1.2 shows an example of abnormal behaviour from the USCD Pedestrian dataset. The problem with anomaly detection in video surveillance is that abnormal behaviours

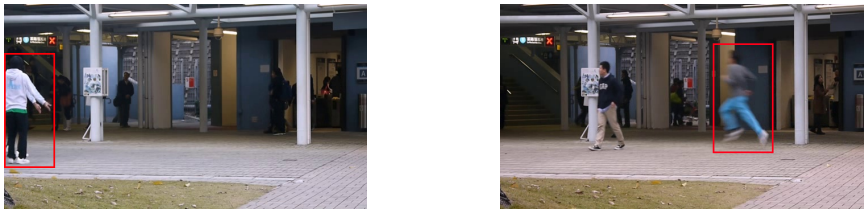


Figure 1.1: Examples of anomalous behaviour in the CUHK Avenue dataset [40]. The anomalies are surrounded by a red rectangle.

Figure 1.2: Examples of anomalous behaviour in the USCD Pedestrian dataset [44]. The anomalies are surrounded by a red rectangle.

are often rare and unbound in anomaly detection applications [39]. As such, it is almost impossible to gather all examples of abnormal behaviour that can occur in a video surveillance application. However, it may be possible to cover and detect more abnormal behaviours by generating pseudo-anomalies that can be added to the training data. The generation of pseudo-anomalies would improve the normal distribution of anomaly detection and theoretically improve the performance of the anomaly detection task. These anomalies would mimic the distribution of the normal data, such as the background and surroundings in the frame. The visualised behaviour in the frame would be different enough to be considered abnormal. Thus the motivation for this project is to explore using generative neural networks to create pseudo-anomalies to improve the performance of anomaly detection, as far as the author knows, has not been attempted before. This motivation can be formulated into a two research questions which will serve as the first basis for the project.

1.1 Initial problem statement

With the introduction and the motivation in mind, two problem statements can be formulated:

1. *How can pseudo-anomalies be generated?*
2. *Can pseudo-anomalies be used to improve the performance of anomaly detection?*

2 Literature Review

The purpose of this chapter is to gain an understanding of the research into artificial neural networks and the development in the field, as well as the current state of the art in the field of generative modelling. First, the chapter will briefly go into the history of artificial neural networks and what makes up a neural network. Then the chapter will describe some of the different types of neural networks used today. Then, two generative neural networks will be described, namely the Generative Adversarial Networks and Diffusion Probabilistic Models. Last, the chapter will conclude with a final problem statement which will be sought answered in the rest of this report.

2.1 Artificial Neural Networks

Research into artificial neural networks has experienced three historical periods of extensive activity. The first peak happened in the early 1940s with McCulloch and Pitts's pioneering work, in which the first model of a neuron was created [30, 38]. The neuron was modelled as a switch which received input from other neurons and was either activated or remained inactive depending on the total weighted input [38]. The second peak happened in the 1960s when Rosenblatt presented the perceptron convergence theorem. Rosenblatt also demonstrated that simple networks could learn from examples, and that a network of McCulloch and Pitts-neurons has similar properties to the human brain. These networks could perform sophisticated pattern recognition and function even if some of the neurons were destroyed [38, 30]. During the second peak, a dampening in the enthusiasm for artificial neural network research happened when Minsky and Paperts showcased the limitation of a simple perceptron. The simple perceptrons were only able to solve linearly separable problems [30]. The third peak happened in the early 1980s when Hopfield's energy approach was presented. In 1986 Rumelhart popularized Werbos' back-propagation learning algorithm, which made it possible to train multi-layer networks [30, 38]. This new back-propagation approach could make fairly complex networks of simple neurons learn from examples, and make them

capable of solving non-linearly separable problems [38]. One of the first early applications of an artificial neural network was NETtalk for machine reading of text [62].

Artificial neural networks, or ANNs, and other types of neural networks, have become popular and helpful models for classification and pattern recognition in a wide field of disciplines. The usefulness of ANNs in machine learning has made them competitive to traditional regression and statistical models. In many fields, such as information security, big data, and cloud computing, artificial intelligence has seen a rise in recent years. With this rise, machine learning, deep learning and ANNs have become a topic of interest [1]. Inspired by the early models of the sensory processing of the brain, ANNs can be created by simulating networks of model neurons in a computer. Through the application of different algorithms that mimic the processes of real neurons an ANN network can learn to solve problems of many types, such as image recognition, natural language processing and image classification [38, 1].

2.1.1 Neurons

What makes ANNs special is the network of artificial neurons that enables the network to learn. But what exactly is a neuron? A neuron is a cell that processes information. A biological neuron is composed of the neucleus (cell body), the axon (transmitter), and the dendrites (receivers) [30]. Within neurons, the nucleus contains hereditary traits and the chemicals needed to produce essential material for the neuron. A neuron receives signals from neighbouring neurons through its dendrites. Electrical impulses are passed from neuron to neuron through synapses when neurotransmitters are released. The electrical impulses is the information that is being processed by the neuron. The effectiveness of a synapse can be adjusted by passing signals, allowing them to learn from the activities they participate in [30]. Each neuron is connected to 10^3 to 10^4 other neurons through roughly 10^{14} to 10^{15} interconnections. Critical information is not directly transmitted between neurons but rather captured in the interconnections [30]. The artificial neurons of an ANN try to emulate this behaviour of biological neurons. In an ANN, a simple neuron is referred to as a threshold unit [38]. This threshold unit is a mathematical function that computes a weighted sum of n input signals and generates an output of 1 if the sum is greater than a threshold value, and 0 otherwise [30]. Mathematically, a neuron can be represented as [3]:

$$y = f(a) = \sigma(\sum_{j=1}^{n} w_j x_j + b) \tag{2.1}$$

where σ is the activation function, in this case, a unit step function, w_j is the weight of the j-th input signal, x_j is the j-th input signal, b is the bias, and y is

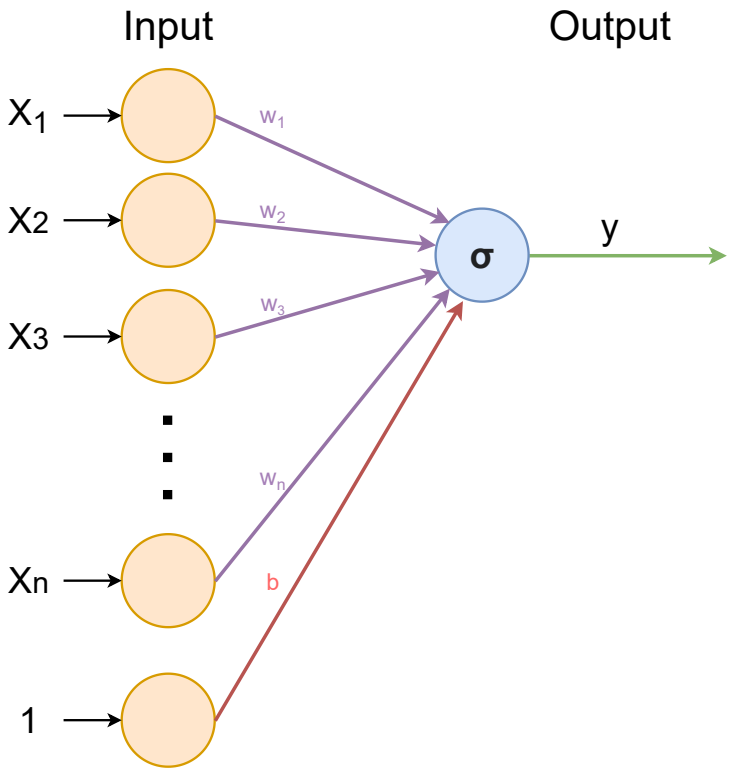


Figure 2.1: Graphical representation of an artificial neuron. A single artificial neuron can contain many different input nodes but only produce one output. All inputs ($x_1..x_n$) and their weights ($w_1..w_n$) are summed together and, alongside a bias (a constant weight), passed through an activation function that produces an output. Adapted from [3].

the output of the neuron [3, 30]. Figure 2.1 shows a graphical representation of the structure of a neuron. As a simplified notation, the threshold u is considered another weight w_0, with a constant input signal $x_0 = 1$ [30]. The positive weights are comparable to the excitatory synapses of biological neurons, and the negative weights are comparable to the inhibitory synapses [30]. In principle, if the weights are suitably chosen, they allow the neurons to perform a wide range of universal computations [30] Jain *et al.*[30] gives a crude analogy of the McCulloch and Pitts artificial neuron to the biological neuron:

- interconnections and wires represent the dendrites and axons.
- the weights represent the strength of the synapses.
- the activation function represents the activity in a cell body.

Due to simplified assumptions, the artificial neuron does not reflect the true behaviour of biological neurons, as the biological neuron is a complex system of many different types of neurons [30]. A way to make the artificial neuron more

accurate is to consider using different activation functions, such as the sigmoid function or a Gaussian activation function [30]. Figure 2.2 shows a graphical representation of different activation functions commonly used in ANNs.

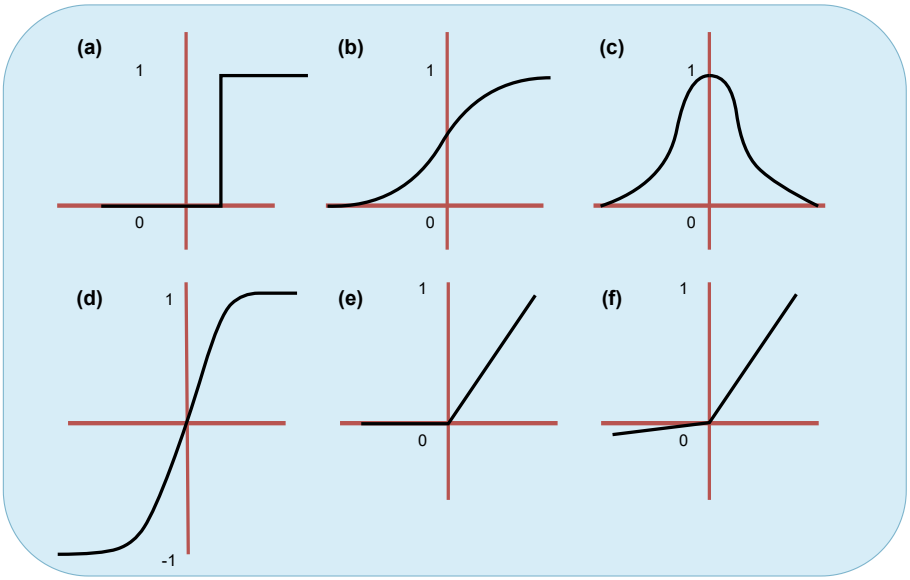


Figure 2.2: Graphical representation of different activation functions. These functions are used to determine the output of a neuron. Each function makes the artificial neuron behave differently. **(a)** Threshold function **(b)** Sigmoid function **(c)** Gaussian function **(d)** tanh **(e)** Rectified Linear Unit (ReLU) and **(f)** Leaky ReLU. Visually adapted from [3] and [30].

2.2 Neural Networks

Neural networks are composed of multiple layers of neurons, where each layer is connected to the next layer. These types of networks have an acyclic structure where the neurons in each layer are fully connected to the neurons in the next layer [3]. The first layer is the input layer, which receives the input signals. The last layer of a neural network is the output layer, which produces the output signals. Any layers in between the input and output layers are called hidden layers. The hidden layers are not directly connected to the input or output layers [3]. Networks with this type of structure are commonly known as Multi-layer Perceptron (MLP), Fully-connected networks and Artificial Neural Networks (ANN) [3]. Figure 2.3 shows a graphical representation of a neural network with three layers, where the input layer has two neurons, the hidden layer has three neurons, and the output layer has one neuron.

Different types of neural networks other than the MLP exists, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). The most

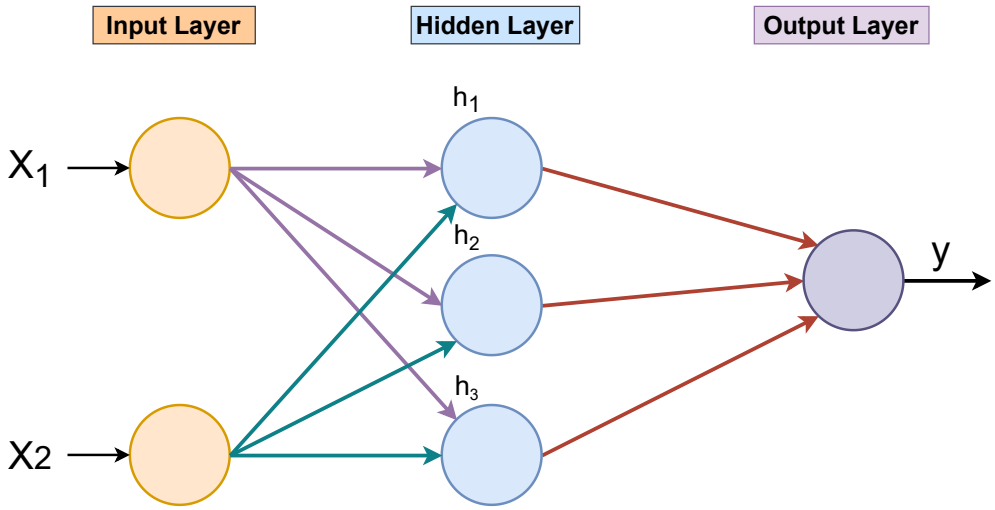


Figure 2.3: Graphical representation of a neural network with three layers. The input layer has two neurons, the hidden layer has three neurons, and the output layer has one neuron. The output of the neurons in the input layer is used as the input to the neurons in the hidden layer. In the hidden layer, the inputs are weighted and summed, and the output is passed through an activation function. The output of the neurons in the hidden layer is used as the input to the neuron in the output layer. In the output layer the output of the entire neural network is computed. Visually adapted from [3].

common types are listed below:

- Perceptron (Single-layer Perceptron)
- Multi-layer Perceptron (MLP)
- Recurrent Neural Networks (RNN)
- Convolutional Neural Networks (CNN)

Another class of neural networks focus on the generative capabilities of neural networks. These types of networks include the two listed below:

- Generative Adversarial Networks (GAN)
- Diffusion Probabilistic Models (DPM)

2.2.1 Common Neural Networks

Perceptron: The simplest form of a neural network is the perceptron [68]. A perceptron contains only an input and output layer, with no hidden layers. It takes a set of inputs, calculates the weighted sum of these, and then applies an activation function to produce the output [68]. Perceptrons are commonly used in linear classification problems, such as the XOR problem, due to their simple architecture [68].

Multi-layer Perceptron (MLP): Also known as feed-forward neural networks, or fully connected networks, are the most common type of neural networks [68]. MLPs are composed of multiple layers of neurons, where each layer is connected to the next layer, meaning the processing power of an MLP is larger than the perceptron. However, the full connectedness of the layers makes the network more complex and prone to overfitting [68]. The problem of overfitting can usually be avoided by using regularisation techniques, such as dropout or early stopping of the training [68]. MLPs have a wide range of applications and are common in data compression, speech recognition and in computer vision applications.

Recurrent Neural Networks (RNN): RNNs are a type of neural network that is commonly used in natural language processing and speech recognition [68]. RNNs are designed to work with temporal or sequential information [68]. These types of networks use data points in a sequence to make a better prediction of the next data point in the sequence. This process is done by taking input and using the activations of the previous or later nodes which influences the output of the current node.

Convolutional Neural Networks (CNN): CNNs are commonly used in image recognition and classification [68]. CNNs are similar to MLPs, but they have a different architecture that is inspired by the visual cortex of animals and humans. CNNs are most commonly used for object detection and classification and are widely used in computer vision applications [68]. What sets CNNs apart from other neural networks is the convolutional layer. The convolutional layer performs a dot product operation between the input and a filter, which is a small matrix of weights. These convolution filters are initialised randomly at the beginning of training, and using loss functions and backpropagation the filters are adjusted to extract features from the input. Training of CNNs is computationally expensive and requires a lot of data to train properly. However, once trained they are very fast at classifying new data. Due to the inspiration from the visual cortex, CNNs are commonly used in image recognition and classification and are the most common type of neural network used in computer vision applications [68].

2.3 Generative Neural Networks

Generative modeling is a type of unsupervised learning, in which the task is automatically learning the patterns or distribution of input data. The learning should be done in such a way that the models can output examples that could have been drawn from the original dataset [8]. In generative modelling two types of neural networks are commonly used: Generative Adversarial Networks and Diffusion Probabilistic Models.

2.3.1 Generative Adversarial Networks:

Generative Adversarial networks (GANs) are a type of neural network that is commonly used in image generation and image-to-image translation tasks [68, 8]. With GANs the generative model is framed as a supervised learning problem, by utilizing two neural networks: a generator and a discriminator [8]. Figure 2.4 shows the training process of a GAN network in graphical form.

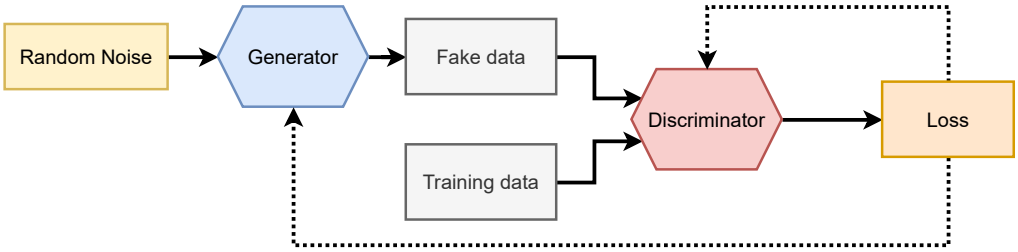


Figure 2.4: Graphical representation of the training process of a GAN network. Random noise is fed into the generator, which synthesizes new data. The discriminator differentiate between the generated data and the training data. Based on the decision loss from the discriminator the generator is updated to improve its performance. Visually adapted from [68].

GANs learn to generate new data by training on a dataset, and then generate new data with the same statistics or distribution as the training data. The generator is trained on the training data and tries to generate new data that is similar to what was seen during training. The discriminator is trained to differentiate between the generated data and the training data, to make a decision of which data is real or fake. These two parts are trained adversarially against each other. The generator tries to fool the discriminator into thinking the generated data is real and uses the decision of the discriminator as an adversarial loss to improve its performance. As training progresses and the two parts of the network improve, the generated data as a result becomes nearly identical in quality to the training data [68, 8].

Application of GANs

Since the first development and emergence of GANs [22], generative models have seen a wide range of applications in recent years. Many GAN approaches exist, especially in the field of image-to-image translation [29, 77, 4, 5, 10, 43, 21, 33], video-to-video translation [66] and text-to-image translation [74]. GANs are also being utilized in anomaly detection due to their adversarial nature [61, 71, 2, 72, 39]. Another field where GANs also can be utilized is open-set recognition and few-shot open-set recognition [36, 53]. Here OpenGAN [36] have shown competitive results compared with other state-of-the-art open-set recognition approaches [36]. In anomaly detection, GANs are used to learn the normal distribution of

data and then use the discriminator to detect anomalies. In GAN-based anomaly detection, the Future Frame Prediction framework [39] has been shown to provide competitive and state-of-the-art results compared to other anomaly detection approaches [39]. Another approach is using GANs to generate pseudo-anomalies and used to train a classifier to detect anomalies. Here the OGNet framework [70] has shown promising results compared with other state-of-the-art anomaly detection approaches [70].

2.3.2 Diffusion Probabilistic Models (DPM):

Diffusion models can be seen as a class of probabilistic generative models learning to reverse a process that has gradually destroyed the training data. This gradual destruction usually comes by gradually adding noise to the training data [11]. The original idea of diffusion models stems from this idea: model a specific distribution from random noise [67]. As such, in the end, the distribution of the generated samples should be as close to the original as possible [67]. Common for all diffusion models is that they share the same baseline processes: Foward and Backward diffusion [67, 11, 14], see Figure 2.5.

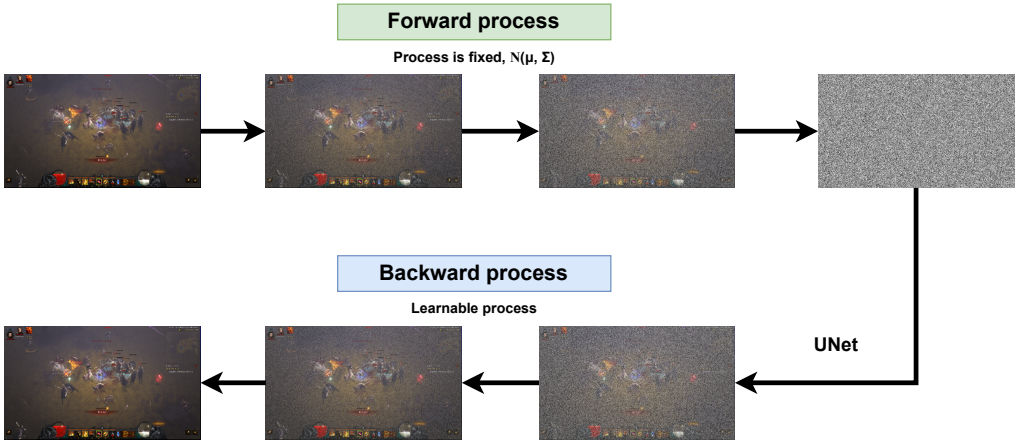


Figure 2.5: Forward and backward process of a diffusion model. During the forward process, noise (often Gaussian noise [14]) is added to the input image at each time step, gradually destroying it using a noise scheduler. The forward process is fixed, and the backward process is a learnable process often utilizing U-Nets [14, 50]. The transition from one latent to another in the latent space is learned at random time steps. Doing sampling all time steps are sampled to reconstruct the input image.

In the forward process, noise is applied, to the images, through a Markov decision process. The added noise usually only depends on the previous image in the training dataset and is sampled using a conditional Gaussian distribution with a mean that depends on the previous image and a fixed variance [14], and is added

to the image by a variance schedule. The variance schedule describes how much noise is added to the image at a specific time step [14, 67]. The reverse diffusion process called reverse diffusion, is where the strength of the diffusion models are - since the goal of the model is to learn the reverse process [67]. The reverse process is learned by training a neural network (usually by using U-Nets [14]) to approximate the probabilities such that the diffusion can be reversed [67]. The backward process starts at a given timestep with Gaussian noise with zero mean and unit variance [14]. Given the current timestep, the transition from one latent to the next latent in the latent space is predicted, making the model learn the probability density of an earlier timestep [14]. During training, the timesteps in the process, are randomly sampled, such that it does not go through the entire sequence of noisy images. At sampling, however, all timesteps are sampled, as the process has gone from pure noise to a final image [14].

Remarks on diffusion model usage

Initially, the idea was to get a baseline of a diffusion model running such that a diffusion model (DM) framework generates the pseudo-anomalies, due to their generative abilities. However, upon researching and testing the topic, it has become apparent that getting a baseline DM running was harder than initially thought, and the time it takes to train a model is longer than thought. Some models have a reported training time between five to seven days [25, 50]. Depending on the amount of testing in the project's experimental phase, it could take more than one week to finish one test. Taking anywhere between three to seven weeks of testing models. That is assuming the first initial test would yield a good result. With the limited timespan for this thesis, it may not be possible to finish on time if a diffusion model has to be trained and tested. Thus, the project will be going in a different direction. The project will instead focus on using Generative Adversarial Networks (GANs) as the generative method for generating the pseudo-anomalies.

2.4 Final problem statement

With the information provided by the literature review, a final problem statement can be made. This statement will serve as the basis for deriving a model architecture in Chapter 3 and serves as the question sought answered in the remaining chapters.

How can a GAN pipeline be designed to generate pseudo-anomalies to improve the unsupervised anomaly detection task?

3 Proposed System

This chapter presents the proposed system, which seeks to answer the final problem statement set in section 1.1. First, the overall architecture will be presented. Afterwards, each subpart of the system is described. The baselines which the proposed system uses will also be presented. The overall purpose of the proposed system is to teach a model to distinguish between normal and abnormal data and predict when a frame in a sequence is abnormal.

3.1 Architecture overview

The proposed system architecture can be seen in Figure 3.1. The system consists of three main parts: A generator architecture, a module for anomaly feature extraction and a module for extracting the normal features. These modules have different tasks.

3.1.1 Generator architecture

The generator architecture generates the pseudo-anomalies used to train the anomaly feature extractor. For this purpose, a GAN architecture will be used, due to their generative abilities. The overall goal of the GAN architecture is to play a mini-max game between the generator and the discriminator to optimize an objective function, to generate the pseudo-anomalies. The GAN architecture is trained on the normal training data of an anomaly detection dataset. Changes to the objective function have been made, such that it does not perfectly reconstruct the normal training data. This change is necessary such that the pseudo-anomalies are different from the normal frames. Once a pseudo-anomaly is generated it is used in the anomaly feature extractor.

3.1.2 Anomaly feature extraction

The anomaly feature extractor is responsible for extracting the features which make up the pseudo-anomalies. In this part of the proposed system, a discriminator is

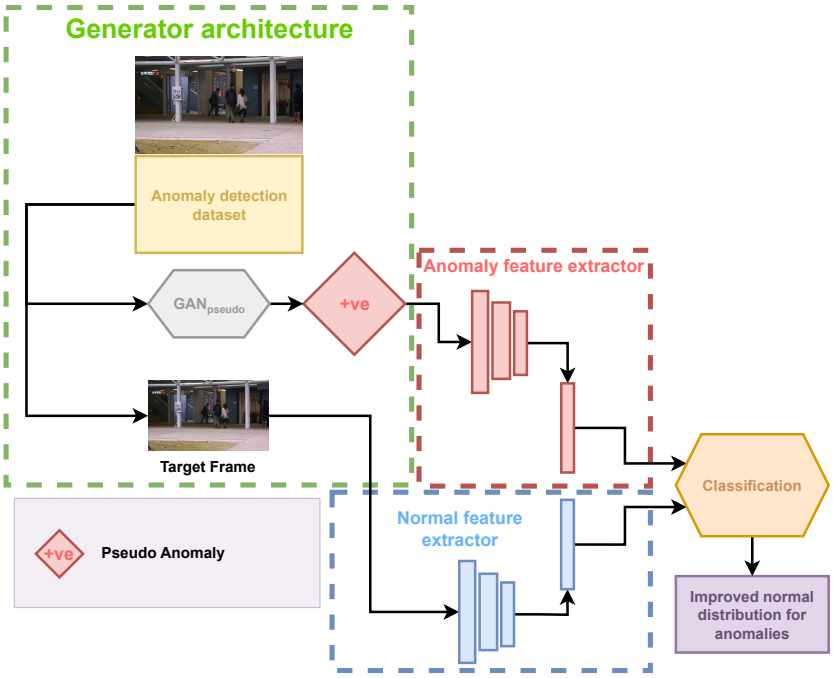


Figure 3.1: The proposed system pipeline. The proposed system consists of four parts: Generator architecture, anomaly feature extraction, feature extraction of normal frames and classification. The generator architecture (green box) is responsible for creating the pseudo-anomalies. These pseudo-anomalies are used in the anomaly feature extractor (red box). The anomaly feature extractor is responsible for extracting the features which make up the pseudo-anomalies. The target frames of the prediction network are fed into the normal feature extractor (blue box), which is responsible for extracting the features which make up the normal frames. These extracted features are used in the classification of normal and abnormal frames.

used. During the training of the anomaly feature extractor, fake anomaly labels are used to train the discriminator. This feature extraction happens in parallel to training the classifier network.

3.1.3 Normal feature extraction

During the training of the GAN architecture, a target frame from the data is used. This feature extractor extracts the features which make up the normal frames. Like in the anomaly feature extractor, fake normal labels are used to train the discriminator. A discriminator is used for this task, which is trained to classify normal frames. This feature extraction happens in parallel to training the classifier network.

3.1.4 Classification

Once the generator architecture can produce pseudo-anomalies, the next step in the proposed pipeline is to conduct feature extraction on the pseudo-anomalies and the target frames. The classification follows the same approach as the discriminator in OGNet [70].

Training the classifier network

The classification network architecture consists of a sequence of convolutional layers, followed by a linear layer and a sigmoid activation function. The linear layer in the classifier network reduces the number of features to two. This linear layer corresponds to the two classes: normal and abnormal. The classification network is trained on the pseudo-anomalies and the target frames. In this regard, fake classification labels are created, where pseudo anomalies classifications are labelled [1, 0], and normal frames are labelled [0, 1]. During training, a binary cross-entropy loss is calculated for both classes. As two losses are applied, the total classification loss is the mean of the two losses added together, which is added to the total generator loss. The calculation of the AUC score is modified to include both a peak signal-to-noise ratio score (psnr score) and a classification score.

AUC-score modification

The AUC score is calculated using the same approach as in Future Frame [39], where the psnr score is used to calculate a normalized score between 0 and 1 for each frame. The psnr score is calculated for each frame in each testing video using the following equation [39]:

$$PSNR(I, \hat{I}) = 10 \cdot \log_{10} \frac{[MAX_{\hat{I}}]^2}{\frac{1}{N}\sum_{i=0}^{N}(I_i - \hat{I}_i)^2} \tag{3.1}$$

A high psnr score indicates that the frame is likely to be normal. The psnr-score is used to calculate a regularity score between 0 and 1 for each frame in the testing videos using the following equation [39]:

$$PSNRs_i = \frac{PSNR_i - \min_i(PSNR_i)}{\max_i(PSNR_i) - \min_i(PSNR_i)} \tag{3.2}$$

This report proposes a modification to the AUC-score calculation, where the classification score is included in the calculation of the regality score. Since the classifier network is trained to classify normal and abnormal frames, the classification score from one index of the classifier output is used to calculate a regularity score between 0 and 1 for each frame using the following equation:

$$Cs_i = \frac{C_i - \min_i(C_i)}{\max_i(C_i) - \min_i(C_i)} \tag{3.3}$$

Where Cs_i is the normalized classification score for frame i, and $\min_i(C_i)$ and $\max_i(C_i)$ is the lowest and highest classification score of frames in a testing video. The normalized scores are then combined to calculate a regularity score between 0 and 1 for each frame using the following equation:

$$S_i = \frac{(Cs_i + PSNR_i) - \min(Cs_i + PSNRs_i)}{\max(Cs_i + PSNRs_i) - \min(Cs_i + PSNRs_i)} \tag{3.4}$$

Based on this score, a prediction of whether a frame is normal or abnormal can be made.

3.2 Baseline Methods

In this section, the baselines used by the proposed system will be presented. Two baselines have been chosen due to their results in anomaly detection. The baselines are: Future Frame Prediction [39] and OGNet [70].

3.2.1 Baseline 1: Future Frame

The Future Frame Framework proposed by Liu *et al.* [39] will be used as a baseline for the generator architecture of the proposed system. It is a GAN baseline for anomaly detection in video sequences based on future frame prediction. The framework proposed by Liu *et al.* [39] can be seen in Figure 3.2.

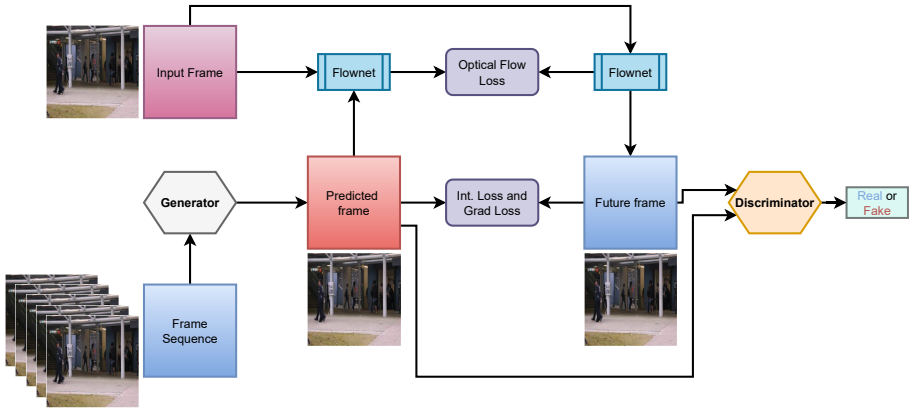


Figure 3.2: The framework proposed by Liu *et al.* [39] for anomaly detection in video sequences. A U-Net generator predicts the next frame in a video sequence. To generate a realistic frame, Liu *et al* applies gradient loss, intensity loss and flow loss to the predicted frame. The flow loss is calculated using a pre-trained FlowNet architecture [17]. During training, an adversarial loss is applied to the discriminator and generator to determine if the predicted frame is real or fake. Visually adapted from [39].

Given a video with consecutive frames, Liu *et al*. sequentially stack frames and feed them into the predictor to predict the future frame. Two constraints are applied to make the predicted frames as close to the real frames as possible: appearance and motion. The appearance constraint follows the same approach as Mathieu *et al*. [46], where the intensity and gradient differences between pixels are used. To preserve the temporal coherence between the predicted frames and real frames, Liu *et al*. [39] utilize FlowNet [17] to estimate the optical flow [39]. This is likewise done for the real future frame and the real frames. The prediction error is calculated using the predicted future frame and the real future frame. Based on this prediction error, a frame is classified as normal or abnormal. For normal frames, the prediction error is low, while for abnormal frames the prediction error is high. The generator architecture is a modified U-Net architecture, in which for each two convolutional layers the output resolution is left unchanged. This means no cropping or resizing is needed when adding shortcuts to the architecture [39]. The GAN framework of this baseline is a Least Squares GAN (LSGAN) [45], to generate more realistic frames. In the proposed system, the generator loss is adapted to generate pseudo-anomalies which lie near-out of the normal distribution from the normal frames.

3.2.2 Baseline 2: OGNet

The generation of pseudo-anomalies will follow a similar approach as the second baseline OGNet [70], chosen due to its results on anomaly generation. The OGNet framework, proposed by Zaheer *et al*. [70], can effectively generate stable results across many training steps. The framework allows for the use of adversarially trained discriminators and generators for robust and efficient anomaly detection [70]. The architecture of OGNet is shown in Figure 3.3. The baseline architecture of OGNet is kept similar to the architecture of [58] to maintain consistency towards other approaches. OGNet is a two-phase training framework. The first phase is similar to common practice adversarial autoencoder training. The second phase optimizes a discriminator by training on several good and bad reconstructions [70]. The good quality reconstructions are provided by the fully trained generator. The bad-quality reconstructions provided by a pseudo-anomaly module and an older generator state [70]. The older state of the generator is the model saved at an earlier epoch than the fully trained generator. In the original implementation of OGNet [70] the older state of the generator is from the first epoch, whereas the fully trained network is at epoch 25.

The pseudo-anomaly module, shown in Figure 3.4, takes care of creating the pseudo-anomalies. The pseudo-anomalies are generated by feeding two arbitrary images into the older state of the generator. These low-level reconstructed images are then meaned on the pixel level. The meaned image is then fed into the gen-

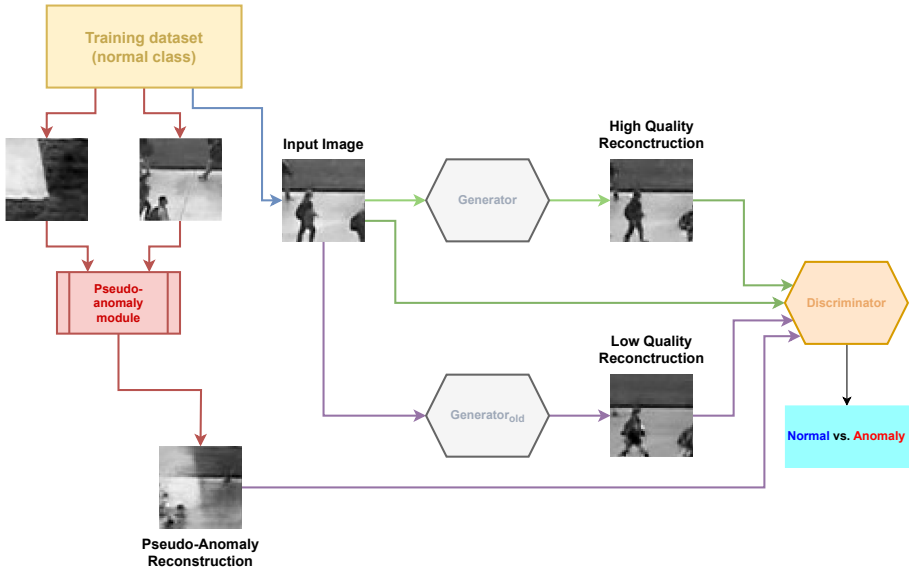


Figure 3.3: The OGNet architecture framework. The baseline training (phase one) is done until a reasonably trained state of the generator and discriminator is achieved. During baseline training, an older state of the generator is stored. During phase two only the discriminator is updated, to distinguish between good and bad-quality reconstructions. Bad-quality reconstructions are provided by a pseudo anomaly module and the outputs from the generator's older state. Visually adapted from [70].

erator to obtain a pseudo-anomaly image. This is to mimic the behaviour of the generator when it gets unusual data.

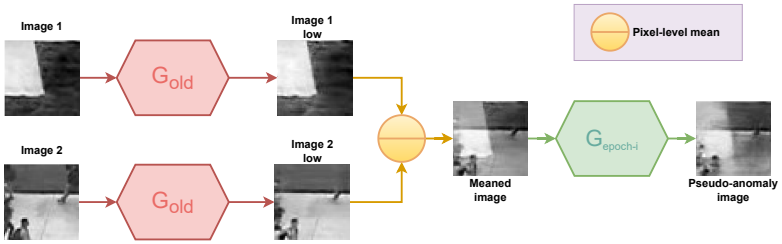


Figure 3.4: The process of generating pseudo-anomalies in OGNet. The pseudo-anomalies are generated by feeding two arbitrary images into the older state of the generator. These low-level reconstructed images are then meaned on the pixel level. The meaned image is then fed into the generator to obtain a pseudo-anomaly image. Visually adapted from [70].

The proposed system will follow a similar idea as OGNet in which a generator generate pseudo-anomalies. Different from OGNet is that the pseudo-anomalies are not created using a pseudo-anomaly module and an older generator-step, instead the anomalies are created alongside the normal frames.

4 Experimental Results

This chapter presents the results of the evaluation of the proposed system from chapter 3. First, the evaluation metrics for the experiments will be presented, including training and the dataset used in the experiments. Then the preliminary experiment with a diffusion model will be presented, followed by baseline experiments using the baselines presented in 3.2. Afterwards, the experiments conducted with the proposed system, starting with experiments for creating the pseudo-anomalies, are presented. Then the quantitative and qualitative results of the proposed system will be presented. In section 4.6, the results of the experiments are discussed. The chapter will end with proposals for any form in section 4.7.

4.1 Evaluation metrics

4.1.1 Performance measurement

The AUC score will be used as the evaluation metric for the proposed system. The score is used to compare the proposed system with the baseline methods. In classification problems, the AUC score is a performance measurement defined as the area under the ROC curve, where the ROC curve is the plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The AUC score represents the degree of separability between the classes, where a higher AUC score indicates a better separability. In the case of anomaly detection, the higher the AUC score, the better the model is at distinguishing between normal and abnormal behaviour. [49]

4.1.2 Dataset

The CUHK Avenue dataset [40] will be used to evaluate the proposed system. This dataset contains 30652 640x360 pixels frames divided between 16 training and 21 testing video clips, captured in an avenue of the Chinese University of Hong Kong (CUHK). The dataset contains 47 abnormal events, which includes running,

throwing objects, and pedestrian walking in the wrong direction. Figure 4.1 shows an example of the anomalies present in the dataset. During training and evaluation, the images are resized to 128×128 pixels.

Figure 4.1: Examples of anomalous behaviour in the CUHK Avenue dataset [40]. The anomalies are surrounded by a red rectangle.

4.1.3 Training the pipeline

The entire pipeline is trained for 10000 iterations, approximately ten epochs, with batch size 16. The GAN architecture is trained using Adam optimisation with a learning rate of $2e^{-4}$ for the generator and $2e^{-5}$ for the discriminator. The classifier is trained using SGD optimisation with a learning rate of $1e^{-4}$.

4.1.4 Sofware

The programming language used for the model is Python 3.8.16, using a created Anaconda Pytorch 1.13.1 environment. The code for the experiments is run on Windows Subsystem for Linux (WSL) [48] with Ubuntu 20.04 as the OS. A GitHub repository has been created to host the code used in this thesis, a link can be found in appendix A.

4.1.5 Hardware

Unless stated otherwise, an NVIDIA GeForce RTX 3060 Laptop GPU with 6GB of memory is used for all experiments.

4.2 Diffusion model experiment

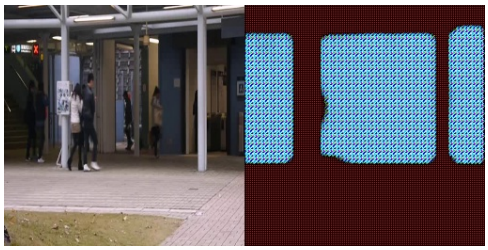
As mentioned in subsection 2.3.2, originally a diffusion model was to be used to generate the pseudo-anomalies. The diffusion model from [50] was tried to get running as a baseline, as the source code for this specific model is publicly available [52]. The model was trained on the CIFAR-10 dataset for 1000 diffusion steps on a cloud server with an available NVIDIA A40 GPU. However, training got stuck after two days of running on training step 780, and as a result, aborted after four days.

As for sampling, the process took a bit shorter, roughly one day, to sample 10.000 images. It took five days total to train the diffusion model. Although it was not a successful experiment, it gave an insight into how long training a diffusion model from scratch can take on a dataset containing low-resolution images (32×32) with basic parameters. The training time would likely have been longer with higher-resolution images, like the ones in CUHK Avenue.

4.3 Baseline experiments

4.3.1 Preliminary auto-encoder result

This test is conducted to see how well a model could generate pseudo-anomalies without adversarial training when using gradient ascent. To this end, a simple auto-encoder is used, following a similar training approach as MNAD [55, 47]. The model was trained for 15 epochs with a batch size of 5, with an initial learning rate of 0.0002 using Adam optimisation. Two types of pixel normalisation are used, one with normalisation between [0, 1] and the other between [-1, 1]. The result can be seen in Figure 4.2. The left images are the input images, and the right is the corresponding generated pseudo-anomalies.



(a) Image normalisation between [0, 1]. The left image is the input image, and the right is the corresponding generated pseudo-anomaly.

(b) Image normalisation between [-1, 1]. The left image is the input image, and the right is the corresponding generated pseudo-anomaly.

Figure 4.2: Pseudo-anomaly by generator without adversarial training.

As can be seen from Figure 4.2a and Figure 4.2b, without adversarial training the pseudo-anomalies generated by the generator are far out of distribution from the training images, which is not desirable as the pseudo-anomalies should only be near out of distribution from the training images. As such, applying adversarial training is necessary to get the generator to generate pseudo-anomalies.

4.3.2 Future Frame Prediction result

As the baseline generator of the proposed system is based on the Future Frame Prediction model [39], tests were conducted to see how well the model predicts

future frames and how well it generates pseudo-anomalies. The model for this test were trained for ten epochs with a batch size of 16 with a generator learning rate of 0.0002 and a discriminator learning rate of 0.00002. Images are resized to size 128×128 and normalised between [-1, 1]. The losses applied to the models are the same as the losses applied in the original implementation of the Future Frame Prediction model [39].

Prediction of future frames

The purpose of this test is to see how well the Future Frame baseline can predict future frames. The result of this test can be seen in Figure 4.3.

(a) Target frame. **(b)** Predicted frame.

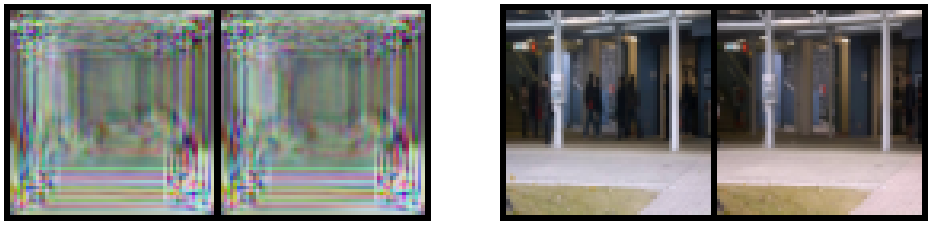
Figure 4.3: Results of the first baseline test of the Future Frame Baseline.

As expected, the model can generate a near-identical image of the target frame, as the same parameters were used in the original implementation of the Future Frame Prediction model [39].

4.3.3 Preliminary OGNet result

The purpose of this test was to check the generative capabilities of the OGNet model [70]. The model was trained for 25 epochs with a batch size of 32, and generator learning rate of 0.001, and a discriminator learning rate of 0.0001. The state of the old generator was chosen as the state the fully trained generator was in after the first epoch. Phase 1 training happened until epoch 20, and phase 2 training (discriminator finetuning) happened until epoch 25. Examples of the generated images can be seen in Figure 4.4a and the corresponding real images in Figure 4.4b.

As can be seen from Figure 4.4a the generated images look mostly like noise and not like the real images in Figure 4.4b. This is most likely because the old



(a) Fake samples from epoch 1. **(b)** Real samples from epoch 1.

Figure 4.4: Preliminary generator results of OGNet. (**a**) Fake samples generated from epoch 1. (**b**) Real samples used to train epoch 1.

generator was chosen as the first epoch which does not provide a good low-quality representation of the input images.

Figure 4.5 shows an example of a generated pseudo-anomaly image during the evaluation of the OGNet model. The top row images are input images, and the bottom row is the corresponding pseudo-anomaly images. In Figure 4.5 it can be seen that the generator can generate something that has a similar distribution as the input image. The generated images are, however, blurry and contain a noise artefact (the white pixels) in the lower right corner. In some cases, the generator can generate a pseudo-anomaly (middle images), although that is not always the case, and those generated are not perfect reconstructions. This is most likely due to the chosen state of the old generator, which does not provide a good low-quality representation of the input images. To provide better results a later epoch for the old generator could have been chosen such as epoch 5. However, as the purpose of this test was to check the generative capabilities ofOGNet, and to get an insight into how pseudo-anomalies can be generated, the results are still useful.

Figure 4.5: Example of the OGNet evaluation of creating pseudo-anomalies. Top row images are the input images during evaluation, and bottom row images are the corresponding pseudo-anomalies constructed.

4.4 Experiments with proposed model

4.4.1 GAN result

The preliminary auto-encoder test made it apparent that adversarial training is needed to generate pseudo-anomalies. As such, another test was conducted to see how well a GAN model could generate pseudo-anomalies. The generator architecture for this test is a standard DCGAN generator [56]. The model for this test is based on MNAD [55, 47] with similar training parameters. The model was trained for 15 epochs with batch size 16, with an initial learning rate of 0.00001 using Adam optimisation. The input images were normalised between [-1, 1] and resized to 128×128. Several tests were also conducted applying more loss functions to the model, using similar losses as [39].

(a) Target frame for reconstruction. **(b)** Reconstructed frame.

Figure 4.6: Anomaly created by generator with adversarial training and standard pixel normalisation of [-1, 1]. The left image is the input image, and the right is the corresponding generated pseudo-anomaly.

The baseline test was conducted using only Mean Squared Error (MSE) loss as the loss function for the generator. The result of this test can be seen in Figure 4.6. The baseline test showed that more constraints on the loss were needed to generate better pseudo-anomalies. For this reason, the intensity, gradient and adversarial losses from [39] are added. The result of this test can be seen in Figure 4.7.

(a) Target frame for reconstruction. **(b)** Reconstructed frame.

Figure 4.7: Anomaly created by a generator using adversarial training and standard pixel normalisation of [-1, 1]. Applying the losses as those in [39], the model can reconstruct something similar to the target frame. The left image is the input image, and the right is the corresponding generated pseudo-anomaly.

While the results from this test gave better results than the baseline test, the generated images still were not reconstructions of the input images. As a final test, increasing the learning rate of the generator from 0.00001 to 0.0002 was tested. The result of this test can be seen in Figure 4.8.

(a) Target frame for reconstruction. **(b)** Reconstructed frame.

Figure 4.8: Anomaly created by a generator with adversarial training, standard pixel normalisation of [-1, 1], and a learning rate of 0.0002. While the generator can reconstruct the background of the image. It is unable to reconstruct the objects in the image.

This test showed that increasing the learning rate improved the reconstruction of the input image. However, it only reconstructed the background. To check if this was due to too few training epochs; a model was trained for 25 epochs using the same learning rate. The test gave the same result as the previous test. This indicates that the model is not learning the objects in the image and that the model only learns the background. Since this problem persists with the DCGAN

generator, the architecture from Future Frame Prediction is used for the generator architecture in the proposed system.

4.4.2 Using KL divergence and perceptual loss

Kullback-Leibler divergence loss (kl-loss) and perceptual loss are added to the model. Kullback-Leibler loss is a measure of disparity between probability distributions and is used to measure the difference between the distribution of the generated image and the target image [32]. The kl-loss is weighted by a factor λ_{kl} to avoid it dominating the other losses. To generate the pseudo-anomalies the kl-loss is maximised. Perceptual loss is used when comparing images that look similar but are not identical, like if the first image was shifted by one pixel in the second image. This type of loss compares discrepancies between images, such as image content and image style [31]. This loss is applied to make sure that the generated pseudo-anomalies are not too far out of distribution from the target image.

For some of the tests, the target images are saved as BGR images, which is why the colour channels are switched in the target images. For all the tests using kl-loss and perceptual loss, the model is trained for 15 epochs with batch size 16, a generator learning rate of 0.0002 and a discriminator learning rate of 0.00002. Images are resized to 128×128 and normalised to [-1, 1].

Experiment 1: 0.1 KL divergence loss

This test was conducted without perceptual loss and with a weight factor of 0.1 for the kl-loss. The result of this test after 15 epochs can be seen in Figure 4.9.

(a) Target frame. **(b)** Pseudo-anomaly.

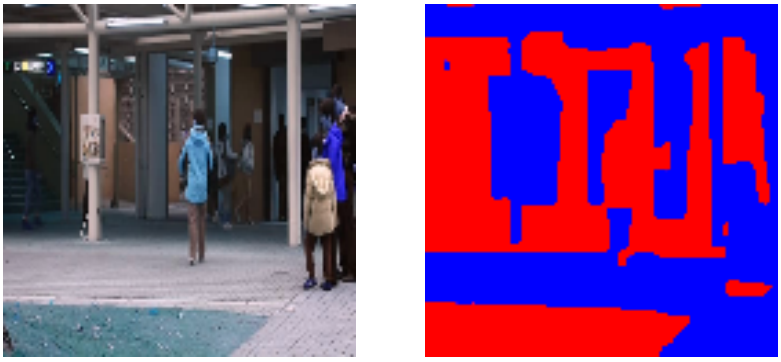
Figure 4.9: Experiment 1: Kl-loss weighted by a factor of 0.1.

From Figure 4.9 it can be seen that with a weight factor of 0.1, the model generates pseudo-anomalies (Figure 4.9b) that are far out of the target distribution

(Figure 4.9a). This is because the KL-loss is weighted too high, as the resulting loss end up becoming 803, while the other losses approach 0, and since the perceptual loss is not applied, it is not keeping the disparity between the image pixels low. This results in the model generating pseudo-anomalies that are far out of distribution from the target image.

Experiment 2: 0.0001 KL divergence loss

Since the kl-loss with a weight factor of 0.1 was too high, another test using a weight factor of 0.001 was conducted. This test was also conducted without perceptual loss. The difference for this test is, during the calculation of the kl-loss the inputs are switched around so that the target image is the input and the predicted frame is the target. The result of this test after 15 epochs can be seen in Figure 4.10.



(a) Target frame. **(b)** Pseudo-anomaly.

Figure 4.10: Experiment 2: Kl-loss weighted by a factor of 0.0001.

While the kl-loss during this test is not as high as the previous test, the kl-loss ends up being 8.14 the model still generates pseudo-anomalies (Figure 4.10b) that are too far out of the target distribution (Figure 4.10a). Once again, this is likely attributed to the fact that the perceptual loss is not applied, and the model is not penalised for generating pseudo-anomalies that are far out of distribution from the target image.

Experiment 3: 0.0001 KL divergence loss with perceptual loss

The main problem with the previous tests was that the model did not have any incentive to generate pseudo-anomalies that were close to the target image. To hopefully fix this problem perceptual loss is added to the model. The perceptual loss is calculated using a VGG19 model using the L1 loss function. The result of this test during the 15 epochs can be seen in Figure 4.11, which shows different steps of

the generated pseudo-anomalies. Figure 4.11b shows the pseudo-anomalies after 150, 300, 450 and 2550 iterations respectively.

(a) Target frames from left to right step 150, 300, 450 and 2550.

(b) Pseudo-anomalies from left to right step 150, 300, 450 and 2550.

Figure 4.11: Pseudo-anomalies when a perceptual loss is applied to the model.

It can be seen that when perceptual loss is applied to the model the model tries to generate pseudo-anomalies that have a similar distribution to the target image. However, even with a perceptual loss, pseudo-anomalies become bluer as the iterations increase, which is likely due to the mode collapse problem that GANs are known to have. Figure 4.12 shows an example of a generated pseudo-anomaly after 15 epochs when the evaluation is run.

(a) Target frame.

(b) Pseudo-anomaly.

Figure 4.12: Experiment 3: Kl-loss weighted by factor 0.0001 and a perceptual loss is applied.

Experiment 4: 0.0001 KL divergence loss with perceptual loss and regularisation

To try and fix the mode collapse problem regularisation is added to the model. To regularise the model, dropout layers have been added after each convolutional layer in the generator. The dropout layers have a dropout rate of 0.5. For this

test, the model was trained for 10 epochs. Figure 4.13 shows different steps of the generated pseudo-anomalies. Figure 4.13b shows the pseudo-anomalies after 100, 300, 1300 and 3900 iterations respectively.

(a) Target frames from left to right step 100, 300, 1300 and 3900.

(b) Pseudo-anomalies from left to right step 100, 300, 1300 and 3900.

Figure 4.13: Experiment 4: Kl-loss weighted by a factor of 0.0001 and perceptual loss and regularisation is applied.

Figure 4.14 shows an example of a generated pseudo-anomaly after 15 epochs during evaluation. Figure 4.14b shows the generated anomalies with perceptual loss and regularisation. Adding the regularisation to the model has not entirely fixed the mode collapse problem, but it has made the mode collapse problem less severe, as the overlaid colour does not increase in intensity. In the discussion chapter of this thesis, it will be discussed why the experiments conducted with kl-loss and perceptual loss did not work as intended.

(a) Target frame.

(b) Psuedo-anomaly.

Figure 4.14: Experiment 4: Kl-loss weighted by a factor of 0.0001 and perceptual loss and regularisation is applied.

4.4.3 Inverting flow loss

The models for these experiments were trained for ten epochs with a batch size of 16, with a generator learning rate of 0.0002 and a discriminator learning rate of 0.00002. Images are resized to 128×128 and normalised between [-1, 1]. Since Future Frame also takes into account the optical flow when generating the next frame, another approach to generate pseudo-anomalies is to increase the flow loss. Inverting the flow loss, in the sense that gradient ascent is applied, will make the model focus more on the optical flow when generating the next frame, thus potentially generating more motion in the image. Another possible outcome of inverting the flow loss is that the model will mimic faulty camera motion, such as a camera that is shaking or the camera losing focus on the object. Three experiments were conducted where gradient ascent is applied to the flow loss. The ascent is applied by multiplying the flow loss by a factor of λ_{flow}. Three experiments were conducted where λ_{flow} is set to 0.5, 0.6 and 0.7.

The kl-loss has been removed from the model, as it is not needed for the experiments, while the perceptual loss is kept such that the disparity between pixels is minimised. As the generated will be closer to the distribution of the target images, the dropout layers have been removed from the generator model.

Experiment 1: 0.5 flow loss

Figure 4.15 and 4.16 shows the result of the first test where λ_{flow} is set to 0.5. Figure 4.15a and 4.16a show the input frames for the prediction network. Figure 4.15b and 4.16b shows the pseudo-anomalies of videos 5, 10, 12 and 21 of CUHK Avenue [40] at iterations 500, 500, 1000 and 0.

Figure 4.15b shows the generated anomalies when the flow loss is inverted by a factor of 0.5. While there is not much difference between the pseudo-anomalies and the input frames, there is a slight difference in the motion between them. The pseudo-anomalies have a slight motion blur, which is not in the input frames. One of these slight differences can be seen in the top right image of Figure 4.15b, where the object thrown by the person is stretched out compared to the input frame in Figure 4.15a. Another difference can be seen in the background of the bottom right image of Figure 4.15b, where a pedestrian is slightly blurred compared to the input frame in Figure 4.15a. The motion blurring is more defined in the top right image of Figure 4.16b, where the pedestrian walking is more blurred compared to the input frame in Figure 4.16a. Comparatively, there is not much difference between the pseudo-anomaly and the input frame in Figure 4.16.

(a) Input frames for the prediction network.

(b) Pseudo-anomlies.

Figure 4.15: Experiment 1: Inverting flow loss by 0.5. The left images are the input frames. The right images are the pseudo-anomalies from the network. Frames from videos 5 and 10 of CUHK Avenue [40].

(a) Input frames.

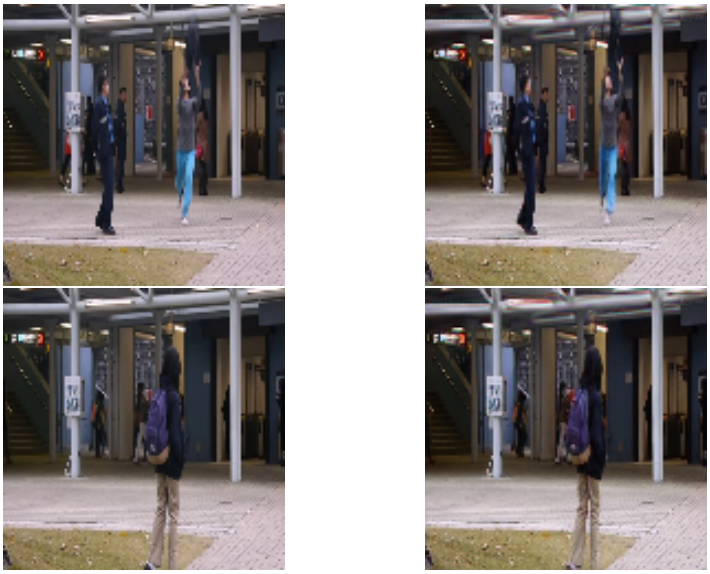
(b) Pseudo-anomalies.

Figure 4.16: Experiment 1: Inverting flow loss by 0.5. The left images are the input frames. The right images are the pseudo-anomalies from the network. Frames from videos 12 and 21 of CUHK Avenue [40].

While the generated frames are not very different from the input frames, it still shows some potential for generating pseudo-anomalies, using the flow loss. The model achieved an AUC score of 81% on the CUHK Avenue dataset.

Experiment 2: 0.6 flow loss

Figures 4.17 and 4.18 show the result of the second test where λ_{flow} is set to 0.6. Figures 4.17a and Figure 4.18a show the input frames for the prediction network. Figures 4.17b and Figure 4.18b show the pseudo-anomalies of videos 5, 10, 12 and 21 of CUHK Avenue [40] at iterations 500, 500, 1000 and 0 respectively. Different from the last test is the images are saved as RGB images instead of BGR images.



(a) Input frames. **(b)** Pseudo-anomalies.

Figure 4.17: Experiment 2: Inverting flow loss by 0.6. The left images are the input frames. The right images are the pseudo-anomalies from the network. Frames from video 5 and 10 of CUHK Avenue [40].

Figure 4.17 shows the result of the second test where λ_{flow} is 0.6, thus inverting the flow loss by a factor of 0.6. The result of this experiment is similar to the first experiment in that the motion blur is similar for the anomalies. One difference in all the images is that with a flow loss of 0.6, it seems like the model can simulate shakiness in the camera, as seen from the fact that the light source at the top of the pseudo-anomalies in Figure 4.17b and Figure 4.18b has moved up and down.

The motion blur on the pedestrian in the top row of Figure 4.18b appears to be more pronounced than in the top row of Figure 4.16b. This, however, is expected, as the flow loss is inverted by a larger factor than in the previous experiment. The

(a) Input frames. **(b)** Pseudo-anomalies.

Figure 4.18: Experiment 2: Inverting flow loss by 0.6. The left images are the input frames. The right images are the pseudo-anomalies from the network. Frames from videos 12 and 21 of CUHK Avenue [40].

motion blur on the pedestrians in the bottom row of Figure 4.18b is similar to the motion blur in the bottom row of Figure 4.16b. The model achieved an AUC score of 78.83% on the CUHK Avenue dataset, which is a decrease of 2.17% compared to the previous experiment. This experiment again warrants further investigation into inverting the flow loss by a larger factor, as it seems like the model can simulate shakiness in the camera. As such, a third test is conducted, where the flow loss is inverted by a factor of 0.7.

Experiment 3: 0.7 flow loss

Figures 4.19 and 4.20 shows the result of the third test where λ_{flow} is set to 0.7. Figures 4.19a and 4.20a show the input frames for the prediction network. Figures 4.19b and 4.20b show the pseudo-anomalies of videos 5, 10, 12 and 21 of CUHK Avenue [40] at iterations 500, 500, 1000 and 0 respectively. Comparatively speaking not much has changed from the previous experiment, as the motion blur is still similar to the previous experiment. The generated shakiness in the camera is no longer present, as the light source at the top of the pseudo-anomalies in Figure 4.19b and Figure 4.20b no longer have the same shakiness as in the previous experiment.

(a) Input frames. **(b)** Pseudo-anomalies.

Figure 4.19: Experiment 3: Inverting flow loss by 0.7. The left images are the input frames. The right images are pseudo-anomalies from the network. Frames from videos 5 and 10 of CUHK Avenue [40].

(a) Input frames. **(b)** Pseudo-anomalies.

Figure 4.20: Experiment 3: Inverting flow loss by 0.7. The left images are the input frames. The right images are the pseudo-anomalies from the network. Frames from videos 12 and 21 of CUHK Avenue [40].

The model achieved an AUC score of 80.40% on the CUHK Avenue dataset, which is a decrease of 0.6% compared to the first experiment.

Pseudo-experiment: Inverting inputs of flow loss calculation

This pseudo-experiment is a byproduct of the testing of the classifier network in section 3.1.4. In general, for this experiment, the inputs to the flow loss calculation are inverted, such that the target frames are used as inputs, and the pseudo-anomalies are used as targets. This was done as the classifier network was not able to distinguish between the target frames and the pseudo-anomalies. The pseudo-anomalies shown in Figure 4.21 are generated during testing of the classifier network after 1000 iterations.

(a) Target frames during iteration 30, 120, 160 and 200.

(b) Pseudo-anomalies during iteration 30, 120, 160 and 200.

Figure 4.21: Pseudo-experiment: Inverting inputs of flow loss calculation. Left images are the input frames, right images are the pseudo-anomalies.

From Figure 4.21 it is seen that the pseudo-anomalies in Figure 4.21b have more motion blur than the frames from the previous experiments and still maintains the distribution of the target frames. With this small pseudo-experiment it is shown that inverting the inputs in the flow loss calculation is a good approach to generate pseudo-anomalies. However, since this is only a pseudo-experiment, the results are not as reliable as the other experiments, as the frames are generated during early iterations of the classifier network and are not taken from the evaluation of the entire network.

4.4.4 Main takeaways

Some of the problems encountered during the experiments will be discussed in chapter 4.6. However, takeaways can be made from the experiments.

Kl-loss

When trying to generate the pseudo-anomalies by applying kl-loss to the network the model was unable to create pseudo-anomalies only near-out of distribution from the targetframes. Adding perceptual loss to the kl-loss in adding more details to the pseudo-anomalies, but the pseudo-anomalies still was too far out of distribution from the target frames. While regularisation often helps with the mode collapse problem of GANs, it did not in this case. Thus, using kl-loss to generate pseudo-anomalies is not a good approach for this problem.

Flow loss

Inversion of the flow loss helped keep the distribution of the pseudo-anomalies closer to the distribution of the target frames. While only three experiments were conducted, it seems that inverting the flow loss to construct the pseudo-anomalies is an approach that shows promise. The author is aware that the pseudo-anomalies generated only contain a small amount of motion blur. The pseudo-experiment inverting the inputs of the flow loss calculation gave better motion blur. However, the pseudo-anomalies generated by the flow loss inversion are still closer to the distribution of the target frames than those created by the kl-loss. As expected when objects or pedestrians are standing still in the target frames, no motion blur is in the pseudo-anomalies. Thus it can be said that using flow loss to generate pseudo-anomalies is a better approach than using kl-loss.

4.5 Evaluating pipeline

In this section, the evaluation of the proposed system is presented. For the remainder of this section proposed system and pipeline is used interchangeably to mean the same thing. The network is evaluated on the evaluation metrics described in section 4.1. The evaluation is performed on the test set of CUHK Avenue and will be compared to the results of other state-of-the-art methods as well as the baseline method Future Frame [39]. The quantitative results compares the proposed system to other state-of-the-art methods in anomaly detection. The qualitative results show how well the pipeline is able to detect anomalies in the test set.

4.5.1 Learning rate test

Two values for the learning rate were tested for the classifier network: 0.001 and 0.0001, to see how a different learning rate affects the AUC score. The results of the test can be seen in table 4.1. The pipeline was tested for 1000 iterations with batch size 16, using the anomaly index of the classification vector.

Table 4.1: Result of testing the pipeline with a classifier learning rate of 0.001 and 0.0001. FF_AUC is the score achieved by the baseline (using a generator with a flow loss inversion of 0.6). P_AUC is the score achieved by the pipeline proposed in this report.

	P_AUC	
Iteration	**500**	**1000**
0.001	56.21%	55.56%
0.0001	67.06%	65.40%

From Table 4.1 it can be seen that the AUC score of the pipeline (P_AUC) increases when the learning rate of the classifier is changed to 0.0001. This is likely due to the classifier network not overfitting as much on the training data when the learning rate is decreased. For testing the pipeline a learning rate of 0.0001 will be used for the classifier network.

4.5.2 Quantitative result

The quantitative results serve as a comparison between the pipeline and other state-of-the-art methods. Table 4.2 shows the result of the proposed systems performance compared to different methods on the CUHK Avenue dataset. The pipeline evaluation using the anomaly classification index is marked P_abnormal, and the pipeline evaluation using the normal classification index is marked P_normal. The generator network is trained with a flow loss inversion of 0.6.

From Table 4.2 it can be seen that the pipeline does not perform as well as the baseline method Future Frame [39] and other state-of-the-art methods (SOTA). Using the anomaly classification index the pipeline performs 16.75% worse than the baseline method, and 9.95% worse than the lowest performing SOTA method ConvLSTM-AE [42]. As the baseline method assumes that normal frames yield a higher psnr score than anomalous frames, the pipeline is also evaluated using the normal classification index. The pipeline improved its performance by 4.07% when using the normal classification index, but it still performs 12.68% worse than the baseline method, and 4.58% worse than the lowest performing SOTA method ConvLSTM-AE [42]. This performance issue is likely due to two facts: (1) the abnormal generator is used as the evaluated generator, and (2) at some point during

Table 4.2: Pipeline result compared to different anomaly detection methods on CUHK Avenue dataset. Ranked from lowest-scoring to highest-scoring. The highest achieved AUC score is marked in bold and underlined, the next highest is in bold.

	AUC
ConvLSTM-AE [42]	77%
Giorno *et al.* [20]	78.3%
Conv-AE [23]	80%
Unmasking [28]	80.6%
Stacked RNN [41]	81.7%
DeepAppearance [64]	**84.6%**
Future Frame [39]	**85.1%**
P_abnormal	68.35%
P_normal	72.42%

the generation of the pseudo-anomalies the frames become too similar to the target frames. These issues will be discussed in section 4.6.5.

4.5.3 Qualitative result

The qualitative results of the proposed system serve as a visual confirmation of how well the pipeline performs the task of anomaly detection. For this purpose, the psnr scores, the classification scores, and the combined scores are plotted for each video in the test set of CUHK Avenue. This will indicate how well the pipeline performs on each video, and if there are any videos that the pipeline performs particularly well or badly on compared to the baseline. Four figures will be presented of videos where parts of the videos contain abnormal frames and two videos where the entire video is composed of abnormal frames.

Detections in videos with both normal and abnormal frames

Video 3: Figure 4.22 shows the the pipeline's performance on video three, which contains both normal and abnormal frames. The ground truth abnormal frames are marked with red boxes. The pipeline performs well in this video. Both the psnr score and the classification score agree with the ground truth, giving lower scores in the abnormal regions and higher scores in the normal regions. Notably is that both the classification score and pipeline start lower than the baseline method, meaning that the pipeline is more inclined to classify the first frames as abnormal. As the video progresses the pipeline and classification score increase, and the pipeline starts to agree with the baseline method. Reaching the first abnormal region, a drop in the pipeline and classification score happens, and they classify

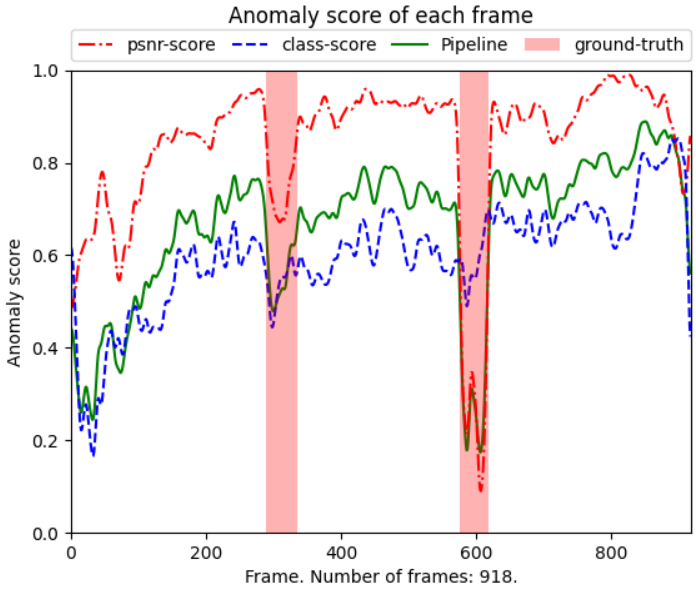
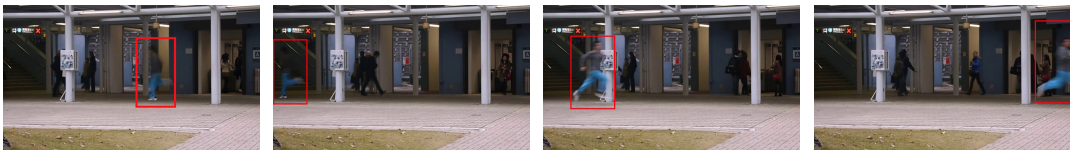


Figure 4.22: Evaluation of the pipeline on video three. The ground truth abnormal frames are marked with red boxes. A larger image can be found in appendix D.

the frames in that region as abnormal. Compared to the baseline method, the drop in the abnormal region is not as pronounced. The classifier is still able to classify the frames in the abnormal region as abnormal for this video. Figure 4.23 shows examples of frames which are successfully classified as abnormal in video threes abnormal region.



(a) Frame 309. **(b)** Frame 335. **(c)** Frame 590. **(d)** Frame 620.

Figure 4.23: Frames from video three's abnormal region which the classifier successfully classifies as abnormal. The anomalies are marked by red boxes.

Video 4: Figure 4.24 shows the proposed system's performance on video four, which contains both normal and abnormal frames. The regions of abnormal frames are marked with red boxes. The classification performs poorly for this video.

It classifies almost all frames as close to normal, while the baseline method classifies the frames in the abnormal regions as abnormal. There is still a small drop in the classification score in the abnormal regions, but it is not as pronounced as

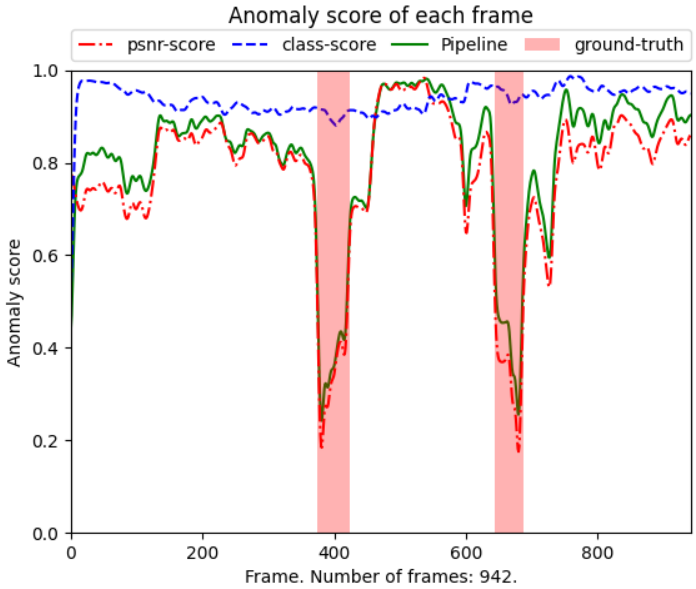


Figure 4.24: Evaluation of the pipeline on video four. The ground truth abnormal frames are marked with red boxes. A larger image can be found in appendix D.

the baseline methods. Figure 4.25 shows examples of frames which the classifier fails to classify as abnormal in video four's abnormal region.

(a) Frame 377: outside first abnormal region.

(b) Frame 385: Failed to detect an anomaly.

(c) Frame 640: outside the second abnormal region.

(d) Frame 660: Failed to detect an anomaly.

Figure 4.25: (a) and **(c)** are frames from just outside the abnormal regions. The classification fails in abnormal region where **(b)** and **(d)** are detected as normal frames.

It can be argued that the frames in Figure 4.25a and Figure 4.25c are classified correctly as abnormal since they are located on the downward curve of the classification score within the region. However, as the scores are sufficiently high, they are classified as normal frames. The difference in performance between video three and four can be due to a lack of motion in the background of the anomalous region of video four. Outside of the anomalous regions of both videos, there is a lot of motion in the background, while the anomalous regions of video four is more static in the sense that only a few background pedestrians are moving.

Video 6: While the classification performed poorly on video four, its performance is still better than on video six. Figure 4.26 shows the pipeline's performance on video six, where the classification classifies all frames, except a few, as normal. As seen in Figure 4.26 the classification score is almost constant throughout the

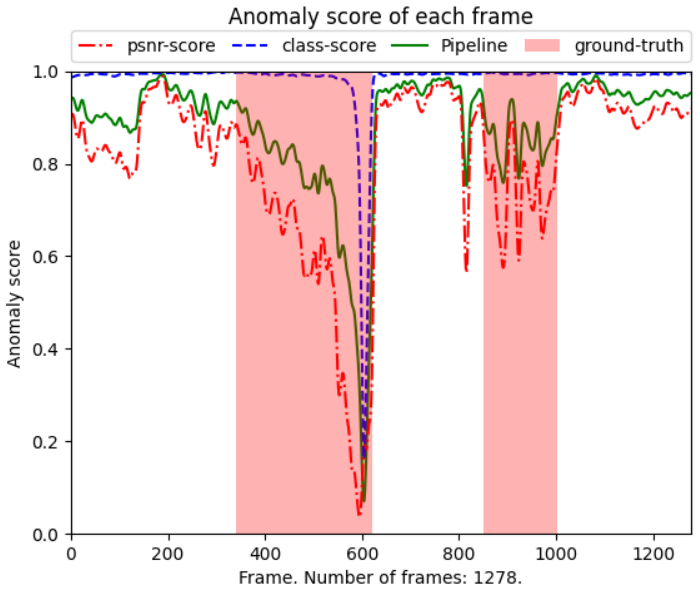


Figure 4.26: Evaluation of the pipeline on video 6. The ground truth abnormal frames are marked with red boxes. A larger image can be found in appendix D.

video, except for a few frames towards the end of the first abnormal region. This performance is likely because there is too much similarity between the normal region and the abnormal region, for the classifier to distinguish between anomalies and normal frames. As the anomaly starts in the background of the video and walks towards the camera, the background is very similar in both the normal and abnormal regions, until the point where the pedestrian is directly in front of the camera. Figure 4.27 shows examples of frames the classifier fails to classify as abnormal in video 6s abnormal region.

(a) Frame 250: outside first abnormal region.

(b) Frame 500: Failed to detect an anomaly.

(c) Frame 600: successful detection.

(d) Frame 950: Failed to detect an anomaly.

Figure 4.27: **(a)** is outside the first abnormal region. **(c)** only successful detection. **(b)** and **(d)** the classification fails in the abnormal region, the frames are classified as normal.

Figures 4.27a and 4.27d are examples of frames in video six which are classified as normal, even though they are in the abnormal regions. This performance issue is likely due to the lack of motion in the background of the video.

Detections on videos with only abnormal frames

The test set of Avenue also contains videos with almost only abnormal frames. Thus it is also fitting to see how the pipeline performs in these videos.

Video 17: Figure 4.28 shows the proposed system's performance on video 17, which contain more abnormal frames than the previous videos. In this video,

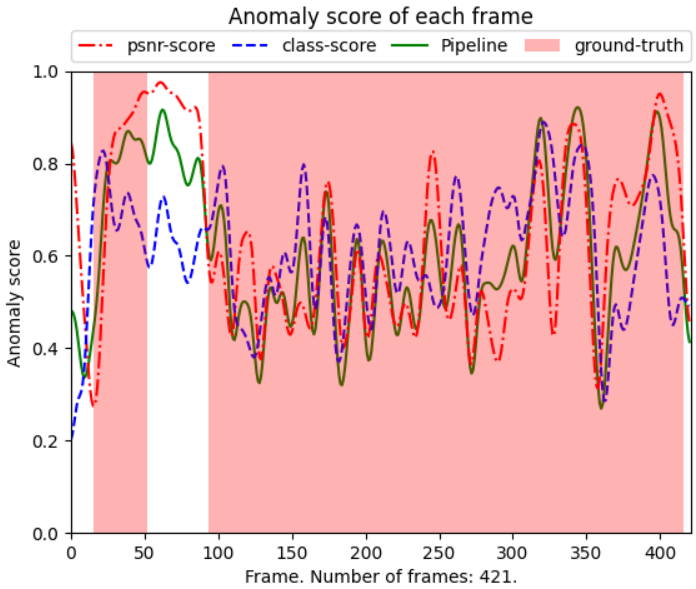


Figure 4.28: Evaluation of the pipeline on video 17. The ground truth abnormal frames are marked with red boxes. A larger image can be found in appendix D.

the classification score and the psnr score are very similar in their detections of abnormal frames. The video starts with only a few normal frames, and then from frame 20 to 50 the video is abnormal. The second abnormal region starts from frame 100 and last until the end of the video. Initially, in the first abnormal region, the classification score is higher than the psnr score, thus the classifier initially classifies the frames as normal. However, along the end of the first abnormal region, the classifier starts to classify the frames as abnormal, where the psnr score is moving towards normal classification. Both the psnr score, and the classification score provide the same classification around frame 30. Figure 4.29 shows examples of frames, and the scores disagree on the boundaries of the abnormal region of video 17. The frame in 4.29a is classified as normal by the classifier, and abnormal

by the psnr score, as the region progresses, the anomaly in frame 4.29b becomes abnormal for the classifier and normal for the psnr score. This means that the

(a) Frame 21.

(b) Frame 51.

Figure 4.29: (a) is classified as normal by the classification score and abnormal by the psnr score. **(b)** is classified as abnormal by the classification score and normal by the psnr score.

classifier makes a mistake initially in the first abnormal region, but corrects itself to see the anomaly as abnormal. Whereas, the psnr score can detect the anomaly as abnormal from the start but then starts to sees it as normal toward the end. In the second abnormal region, the psnr score and the classification score are in agreement on the classification of most of the frames, but at some points, they disagree. Figure 4.30 shows examples where the scores disagree and agree on the classification. Specifically, for this video, it is hard to determine what makes the

(a) Frame 190. Both scores agree.

(b) Frame 353. Both scores agree.

(c) Frame 160. Scores disagree.

(d) Frame 290. Scores disagree.

Figure 4.30: (a) and **(b)** both scores agree on the abnormal classification. **(c)** and **(d)** The scores disagree: Frame classified as normal by classified, abnormal by psnr score.

classification score and the psnr score disagree. A factor that could be the cause is how the movement of the anomaly was detected by both scores. Which could be attributed to the pseudo-anomalies generated during the training of the classifier.

Video 18: Figure 4.31 shows the pipeline's performance on video 18. For the first part of the video in Figure 4.31 up until frame 150, the classification score can better distinguish between normal and abnormal frames than the psnr score. However, after frame 150, the classification score increases to the same level as the psnr score. Where the classification is more successful than the psnr score is around frame 100, where the classification score is lower than the psnr score. At frame 100 the classifier determined that the frame is likely more abnormal than

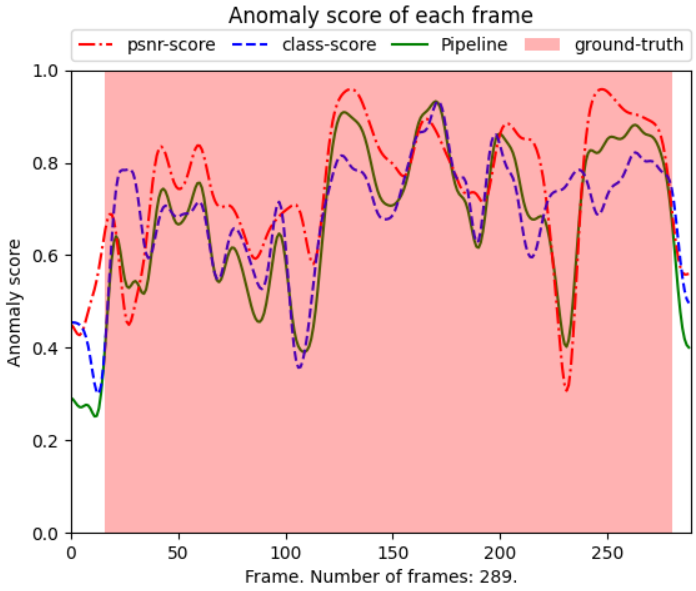
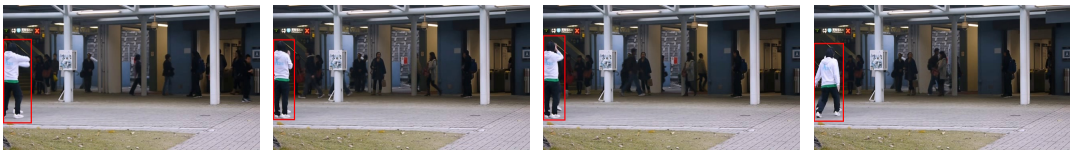


Figure 4.31: Evaluation of the pipeline on video 18. The ground truth abnormal frames are marked with red boxes. A larger image can be found in appendix D.

normal, while the psnr score did not. However, at frame 225, the psnr score is lower than the classification score. Thus the psnr score determined the frame to be more abnormal, while the classifier did not. On most parts of the video, both scores agree on the classification of the frames, except for a few frames. Figure 4.32 shows examples of frames on which the classifier and psnr score disagree. The pattern



(a) Frame 26. **(b)** Frame 210. **(c)** Frame 225. **(d)** Frame 248.

Figure 4.32: (a): Classification score is higher (normal) than psnr score (abnormal). **(b)**: classification score lower (abnormal) than the psnr score (normal). **(c)**: classification score much higher (normal) than the psnr score (abnormal). **(d)**: classification score lower (abnormal) than the psnr score (normal). Anomaly is marked with a red box.

once again seems to be that the classifier is more successful at detecting anomalies when there is motion in the background of the video, except for some outliers like frame 225.

Video 21: Figure 4.33 shows the pipeline's performance on video 21. In this video, the psnr score is more successful at detecting anomalies than the classification score. As seen, the classification score fluctuates a lot more than the psnr score, giving a lot of frames at the beginning of the video a high classification score, even though they are abnormal. Halfway through the video, the classification score can better distinguish between normal and abnormal frames, but still not as good as the psnr score. It peaks towards the end of the abnormal region. The result of this video contrasts with the result of video 18, where the classification score was able to better distinguish between normal and abnormal frames.

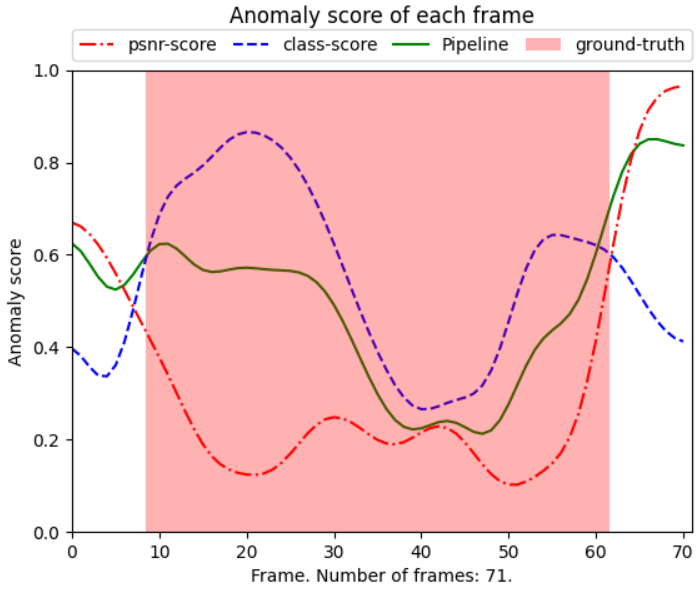


Figure 4.33: Evaluation of the pipeline on video 21. The ground truth abnormal frames are marked with red boxes. A larger image can be found in appendix D.

Figure 4.34 shows examples of frames in which the classifier gives a score close to normal, and the frames it gives a score closer to abnormal. Where the classifier

(a) Frame 20: Normal classification (failure).

(b) Frame 40: Abnormal classification (success).

(c) Frame 55: Normal classification (failure).

Figure 4.34: (a): Frame classified as normal. **(b)**: Frame classified correctly. **(c)**: Frame classified wrongly. Example anomaly is marked with a red box.

fails to detect the anomaly in frame 20, it can detect the anomaly in frame 40.

However, it fails to detect the anomaly again in frame 55. This pattern seems to contradict the assumed that the classifier is more successful at detecting anomalies when there is motion in the background of the video. However, comparing the foreground in Figure 4.34 with the foreground in Figure 4.32, the foreground is also more complex in the sense more is happening in the image.

4.5.4 Main takeaway from the qualitative results

It seems to be that the classifier is more successful at detecting anomalies when there is more motion in the background of the videos. When the motion in the background is different from the motion in the foreground, the frame is more likely to be classified as abnormal, and the classification is assigned a low score. Thus, when there is little to no motion in the background of the video, the classifier is more prone to fail at detecting anomalies, as they become the "normal" motion in the video. It also happens when the anomaly is first in the background and then moves to the foreground, as seen in Figure 4.27. The less complex the foreground is, the better the classifier most likely is at detecting anomalies. As could be seen in Figure 4.28 at certain frames during the video, the classifier gives a slightly better anomaly score than the psnr score. This could be because the classifier has been trained on the pseudo-anomalies. As for the pipeline as a whole, while it did not perform as well on the AUC evaluation as the baseline, it can detect anomalies in certain videos where the baseline is not able to detect them. This is due to how the pipeline score is calculated, as it is a combination of the psnr score and the classification score. Thus, the higher both scores are, the higher the pipeline score is.

4.6 Discussion of results

4.6.1 Vanishing foreground objects in GAN architecture

As mentioned in section 4.4.1, the GAN architecture encountered a problem with vanishing foreground objects. It is still uncertain what caused this problem, as it was able to generate the background of the frames almost perfectly. One possible cause of the problem could be some form of mode collapse or a problem with vanishing gradients during training.

4.6.2 Problems with kl-loss application

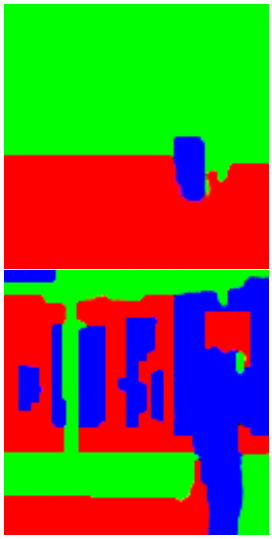
As mentioned in subsection 4.4.2, applying kl-loss to the generator network generated frames with distributions far outside the distribution of the input frames. While adding perceptual and dropout to the generator network helped to reduce the problem, it did not solve it completely. The above hints at the kl-loss not being

a feasible loss function for generating frames with a GAN architecture, as it seems like the kl-loss works on the histogram of the coulour channels and not the actual frames. This might be because kl-loss is a method for comparing data distributions and not actual frames.

Results from other tests

Aside from those presented in subsection 4.4.2, a few other tests were conducted. These tests were to see if the training epochs affected the kl-loss. However, they still showed the same results as the tests presented in subsection 4.4.2. The result from the test is seen in Figure 4.35. Figure 4.35a shows the input images for the generator at epoch two and epoch five respectively, while Figure 4.35b shows the generated pseudo-anomalies. Different epochs still showed the same result,

(a) Input frame for the generator at two epochs (top) and five epochs (bottom).



(b) Generated pseudo-anomaly at two epochs (top) and five epochs (bottom).

Figure 4.35: Generator results from testing for kl-loss. Results from two and five epochs.

with the generator frames being far outside of the normal distribution of the input frames. However, as seen in Figure 4.35b, more objects of different colours are in the background of the pseudo-anomaly. It is trying to generate the positions in the image where objects are present (visualized by the blue blobs), but it is not generating the objects themselves.

4.6.3 Inversion the flow loss

The idea behind inverting the flow loss came while trying to figure out how pseudo-anomalies could be created without deviating too much from the distribution of the target images. This idea was based on the fact that the original implementation of Future Frame Prediction [39] uses the flow loss to predict future frames by taking into account the motion of objects. By wrongly estimating the motion of objects, pseudo-anomalies could be generated. Thus the idea of inverting the flow loss was born, which as shown in subsection 4.4.3 had some potential, even if only minor differences between input and output frames were observed.

Larger inversion factors

As only a few inversion factors were tested, possibly larger inversion factors could have generated better results, as there were not many differences between the inversion factors tested. To test this, a few more inversion factors were tested, which can be seen in Figure 4.36. These were not trained for the full amount of iterations as the other inversion factors, as it was only to see if larger inversion factors would generate better results.

(a) Inversion factor 1.1.

(b) Inversion factor 1.3.

(c) Inversion factor 1.5.

Figure 4.36: Generator results from testing for inversion factor. The top image is the input frame for the generator. The bottom images are the generator frame. Results from 1.1 and 1.2 and 1.5.

The images in Figure 4.36 (4.36a,4.36b,4.36c) show that larger inversion factors are better at generating more motion blur in the images while they still maintain the distribution of the input frames. The AUC score of these inversion factors is also quite similar to the AUC score of the other experiments. Inverting the flow loss by 1.1 generated an AUC score of 81.66%. Inverting the flow loss by 1.3 generated an AUC score of 81.7%. Inverting the flow loss by 1.5 generated an AUC score of 81.75%. A further invesigation on larger inversion factors could be made in the future, to determine how large the factor can be.

Issue when inverting the flow loss

While not entirely a big issue, it is still worth mentioning that when training the flow loss with an inversion factor, while the loss starts negative, it still goes towards zero. The below output is an example from training the flow loss with an inversion factor of 1.3, where the flow loss increase towards zero.

Listing 4.1: Example of the loss when inverting the flow loss (fl_loss). Other outputs have been removed to make the example more readable.

```
[10]   fl_l: -1.123 | gan_l: 0.131 | G_l_total: 8.471
[50]   fl_l: -0.565 | gan_l: 0.219 | G_l_total: 4.267
[100]  fl_l: -0.590 | gan_l: 0.231 | G_l_total: 3.763
[150]  fl_l: -0.660 | gan_l: 0.328 | G_l_total: 2.464
[200]  fl_l: -0.550 | gan_l: 0.302 | G_l_total: 2.328
[250]  fl_l: -0.425 | gan_l: 0.282 | G_l_total: 2.183
[300]  fl_l: -0.383 | gan_l: 0.267 | G_l_total: 2.685
```

Meaning that the longer the model is trained, the less the flow loss is taken into account. Inverting the input and output when calculating the flow loss was tried as a solution to this, but it did not change the results. As it increases towards zero, the flow loss might be taken into account less and less, which could be the reason why the generated frames do not have as much motion blur as expected after each iteration. However, as seen from the test results, both in the experimental results and the results from this section, motion blur is still added to the generated frames.

4.6.4 Classifier

Initially, the idea was to use pre-trained ResNet18 networks as the classifier and finetune the last layer to the pseudo-anomalies and the target frames. However, issues with the pre-trained ResNet18 network occurred when trying to finetune it. Only finetuning the last layer of the ResNet18 network resulted in the classifier predicting the wrong labels for the frames. Alongside this, an issue with the loss occurred, where the loss would go towards zero after only a few iterations. The issue with the wrong classification was solved by finetuning the weights of all layers

in the ResNet18 network, resulting in the classifier predicting more correct labels. The loss issue was not solved entirely but limited by using the same learning rate as the discriminator of the GAN network. This fix resulted in the loss not decreasing as fast, but it reached zero after a couple of hundred iterations. These issues were likely caused by the pre-trained network overfitting the training data. Instead of using the pre-trained ResNet18 networks, it was decided to use a modified discriminator architecture of OGNet [70] and train the classifier from scratch. Using this discriminator the issue with the loss disappeared, and the classifier was able to learn.

4.6.5 Results of pipeline evaluation

Quantitative results

As could be seen in table 4.2 the proposed system at its current stage is not able to provide competitive results to other anomaly detection approaches. Comparing it to the baseline it was not at all able to compete. This issue is due to how the pipeline is constructed and the evaluation is conducted. First, since the evaluation requires a generator in the evaluation, the pseudo-anomaly generator had to be evaluated. Since the generator is adversarially trained, it tries to generate images near out of the distribution of the target images. The psnr score is calculated using the pseudo-anomalies and the target images. If the pseudo-anomalies are different from what is assumed to be the target, the psnr score will be lower by default. This results in a low AUC score.

(a) Iteration 100. Left: Normal frame. Right: Generated pseudo-anomaly.

(b) Iteration 700. Left: Normal frame. Right: Generated pseudo-anomaly.

Figure 4.37: Examples of generator input and outputs at different iterations during training of the entire pipeline. The generator starts to generate frames that are too similar to the normal frames, which makes it difficult for the classifier to distinguish between normal and abnormal frames.

Another issue that might have caused the result, is the fact that at some point during the generation of the pseudo-anomalies, they become too similar to the normal frames, with only slight differences. An example of the pseudo-anomalies generated during the evaluation is seen in Figure 4.37 and Figure 4.38.

In Figure 4.37 the pseudo-anomalies look different from the normal frames, in the sense that they are more blurry and have a slightly different colour scheme. This difference is especially seen in Figure 4.37a. Where in later iterations, the pseudo-anomalies become too similar to the normal frames, with only slight differences, as seen in Figure 4.38. This issue happens as the flow loss increase towards zero (as explained in Section 4.6.3) and likely causes the classifier to classify normal frames as pseudo-anomalies, which would result in a lower AUC score. It should be noted that the evaluation of the pipeline was on a model with an inversion factor of 0.6. As such, the pseudo-anomalies only had minimal differences from the target frames. Thus, to further improve the pipeline, it is suggested to increase the value of inversion factor to make the pseudo-anomalies more different from the target frames (as documented in section 4.6.3).

(a) Iteration 5400. Left: Normal frame. Right: Generated pseudo-anomaly.

(b) Iteration 10000. Left: Normal frame. Right: Generated pseudo-anomaly.

Figure 4.38: Examples of generator input and outputs at different iterations during training of the entire pipeline. The generator starts to generate frames that are too similar to the target frames, which makes it difficult for the classifier to distinguish between normal and abnormal frames.

Qualitative results

Section 4.5.3 went through the qualitative results of testing the baseline and the proposed system from chapter 3. The results showed that both the psnr score and the classification score agreed on the classifications of frames and disagreed on others. As seen in figures 4.28 and 4.22, the two scores primarily agreed on the classification of the frames, and in figures 4.24 and 4.26 the two scores disagreed on the classifications. Since the pipeline is a combination of the psnr score and the classification score, the pipeline is more certain of the classification of the frames that both the psnr score and the classification score agreed on. Thus on certain videos the pipeline gives better AUC scores. However, in evaluating the two scores individually, the classifier only yielded a total AUC score of 55.46% which is similar to randomly guessing the classification of frames. While the baseline scored a total AUC score of 79.26%, which is lower than the official AUC score of the original implementation of [39]. Since the pseudo-anomaly generator used in the evaluaion, the psnr score calculated during the qualitative results are for

the pseudo-anomalies. The psnr score is thus not calculated for the target frames, which also lowers the AUC score.

Improvement to the proposed system

To improve the AUC score, a two-branch generator architecture is needed. In the first branch, a generator takes care of generating the pseudo-anomalies. This branch follows the same architecture as the currently proposed system and is trained in the same way. The purpose of this branch is to get the classification score used in the evaluation of the pipeline.

In the second branch a generator takes care of generating the predicted frames. This branch is trained the same way as the first baseline [39] described in section 3.2.1. The purpose of this branch is to generate the psnr score used in the evaluation of the pipeline. Also, with this branch, the classifier is trained on the predicted frames instead of using the target frames. This update to the proposed system can be seen in Figure 4.39.

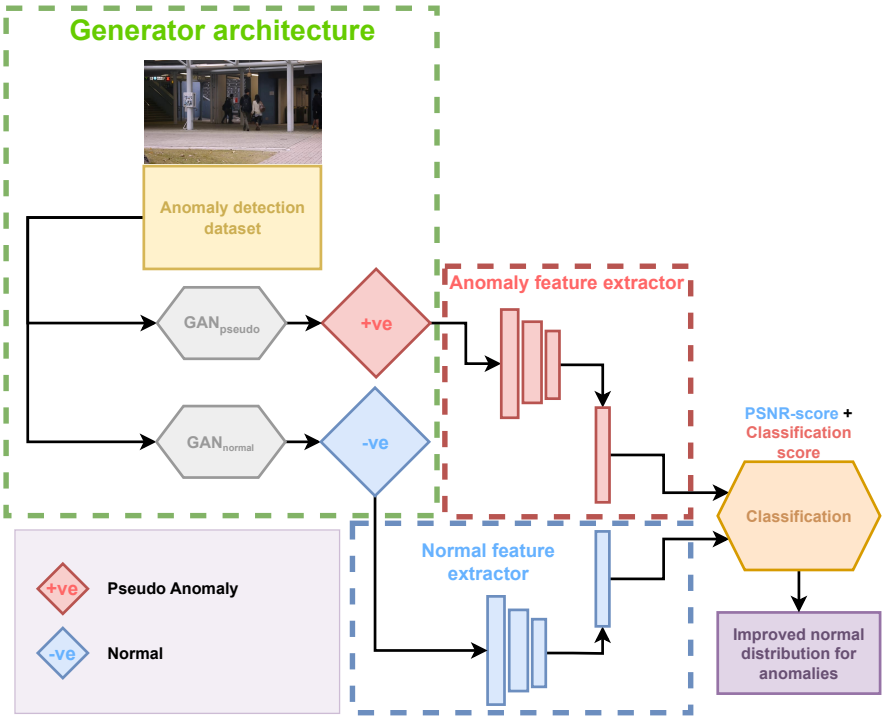


Figure 4.39: Updated proposed system with a second generator branch. The first branch generates the pseudo-anomalies, and the second branch generates the predicted frames. The classifier is trained on both the generated pseudo-anomalies from the first branch and the predicted frames from the second branch.

Both branches of the generator are jointly trained, and the classifier is trained on both the pseudo-anomalies and the predicted frames.

4.7 Future work

For future work, a few things could be considered. First of all, improving the architecture by adding the second branch of the generator as described in section 4.6.5 would take priority. This change to the proposed system would improve the AUC score and make the pipeline more competitive. Secondly, the system should also be evaluated on other anomaly detection datasets such as the USCD Ped1 and Ped2 datasets [44] and Shanghai Tech dataset [76]. Evaluating the proposed system on other datasets would give a better understanding of the performance. It would also make it possible to compare the proposed system to other anomaly detection methods, including the second baseline OGNet [70], as OGNet is only evaluated on the Ped 2 dataset. Being able to compare the proposed system with OGNet would be ideal as the pseudo-anomaly generation of the pipeline is in part inspired by the pseudo-anomaly generation of OGNet.

Another thing that could be investigated further is the use of the flow loss. It was briefly shown in the discussion of the results that increasing the inversion factor of the flow loss generates pseudo-anomalies with more motion blur and fewer details, than the inversion factor used in the evaluation of the proposed system. Potentially by increasing the inversion factor of the flow loss, the pseudo-anomalies would be more different from the normal frames, and thus the classifier might be able to find a better decision boundary between normal and abnormal frames. This might also help improve the AUC score of the pipeline

5 Conclusion

Due to the rare occurrence of anomalous scenes, anomaly detection is often seen as a one-class classification and novelty detection problem. Anomalous scenes are often not well represented in video surveillance applications and it can be almost impossible to collect enough representative data. The application of pseudo-anomalies could improve the performance of anomaly detection algorithms. With the above in mind, this thesis initially set out to answer the following research questions:

- *How can pseudo-anomalies be generated?*
- *How can pseudo-anomalies be used to improve the performance of anomaly detection algorithms?*

To answer the research questions a literature review was conducted to get an understanding of artificial neural networks and their different architecture types. As generating pseudo-anomalies is a task of generating new data, the literature review also covered generative methods like GANs and diffusion models. Deriving from the literature review, that GAN provides a good framework for generating new data, a final problem statement was formulated:

How can a GAN pipeline be designed to generate pseudo-anomalies to improve the unsupervised anomaly detection task?

With the problem statement in mind, a GAN pipeline was designed and implemented with the overall purpose of teaching a model to distinguish between normal and abnormal scenes and predict when a frame in a sequence is anomalous. The pipeline consists of three modules: a generator architecture, an anomaly feature extractor and a normal feature extractor. The generator architecture is trained to generate pseudo-anomalies, the anomaly feature extractor extracts features from the pseudo-anomalies and the normal feature extractor extracts features from normal frames. A classifier is trained on the extracted features to distinguish between normal and abnormal frames. The pipeline is inspired by two promising baseline methods: Future Frame Prediction [39] for the anomaly detection task and OGNet

[70] for generating pseudo-anomalies.

Experiments were conducted on the generator architecture to find the most feasible way to generate pseudo-anomalies. Preliminary experiments using an auto-encoder as the generator architecture showed that adversarial training was needed to generate pseudo-anomalies. Testing the baseline methods proved this to be true as well. The experiments conducted with the proposed model sought to find the best way to generate pseudo-anomalies. Initially, a DCGan architecture was used as the generator architecture for generating pseudo-anomalies. However, the DCGAN architecture was only able to generate the background of the surroundings in the training images, not moving objects. It is speculated that this is due to vanishing gradients or the problem of mode collapse. The DCGan architecture was therefore replaced with a U-Net architecture.

Using a U-net architecture the generator was able to generate moving objects in the scene. To generate the pseudo-anomalies the generator was trained with a combination of the losses used by a baseline method and kullback-leibler divergence loss. However, using kullback-leibler divergence loss the pseudo-anomalies were too far out of distribution from the training data, as it is likely that kullback-leibler loss is only working on a histogram of frames. The generator architecture was instead trained using the losses introduced by the baseline method and inverting the flow loss. With this setup, the generator was able to generate pseudo-anomalies from the training data. A larger inversion of the flow loss shows promise as more motion blur is added to the pseudo-anomalies. Alongside the generator architecture, a classifier was trained to distinguish between normal and abnormal frames.

The entire pipeline was evaluated on the CUHK Avenue dataset in which quantitative and qualitative results were assessed. The quantitative results were a comparison between the proposed system and state-of-the-art methods of anomaly detection using the AUC score as the evaluation metric. For the proposed pipeline the AUC score was modified to include both a classification score and a psnr score. Evaluating the pipeline on the anomaly classification index of the classification vector, the proposed pipeline achieved an AUC-score of 68.35%, 16.75% lower when compared to the baseline method. Evaluating the pipeline on the normal classification index of the classification vector, the proposed pipeline achieved an AUC-score of 72.42%, 12.68% lower when compared to the baseline method. These low-performance scores are likely due to two reasons:

1. The pseudo-anomaly generator is the generator evaluated when calculating the psnr scores.
2. At some point during the generation of the pseudo-anomalies the frames become too similar to the target frames.

The qualitative results served as a visual confirmation of how well the pipeline performed the task of anomaly detection. The classification score, psnr scores and ground-truth labels were combined in a plot to visualize the performance of the pipeline on each video in the testing set of CUHK Avenue. These results showed that the pipeline, in some videos, was able to detect the same anomalous frames as the baseline method, whereas in others the pipeline was not able to detect any anomalous frames. It is speculated that the pipeline is not able to detect anomalous frames in videoes where there is little movement in the background scene.

While the pipeline, at its current stage, is not able to provide competitive results, it is still believed that introducing pseudo-anomalies to the anomaly detection task is a promising approach. To improve the performance of the pipeline, it is proposed for future work to introduce a second branch to the generator architecture. This branch is to be trained to generate normal frames, to calculate a more correct psnr score.

Future work on the pipeline can be summarized as follows: Firstly, improving the pipeline by introducing a second branch to the generator architecture takes priority. However introducing a second branch to the generator architecture will likely require a larger GPU, than the one used, in terms of memory. Secondly, the pipeline should be evaluated on other anomaly detection datasets, such as USCD Ped2 [44] and ShanghaiTech [76]. Lastly, it should be investigated how large the inversion factor of the flow loss can be before the pseudo-anomalies generated are too far out of distribution from the training data.

Bibliography

[1] Oludare Isaac Abiodun et al. "State-of-the-art in artificial neural network applications: A survey". In: *Heliyon* 4.11 (2018), e00938. ISSN: 2405-8440. DOI: https://doi.org/10.1016/j.heliyon.2018.e00938. URL: https://www.sciencedirect.com/science/article/pii/S2405844018332067. (Retrieved: 2023-04-05).

[2] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. *GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training*. 2018. arXiv: 1805.06725 [cs.CV]. (Retrieved: 2023-05-21).

[3] Chris Holmberg Bahnsen. "An introduction to artificial neural networks - part I". Introduction lecture covering basics of artificial neural networks. 7th Semester Vision Graphics and Interactive Systems, Aalborg University, Denmark. 2022-03-21. (Retrieved: 2023-04-24).

[4] David Berthelot, Tom Schumm, and Luke Metz. "BEGAN: Boundary Equilibrium Generative Adversarial Networks". In: (Mar. 2017). (Retrieved: 2023-02-12).

[5] Vineeth S. Bhaskara et al. "GraN-GAN: Piecewise Gradient Normalization for Generative Adversarial Networks". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 2432–2441. DOI: 10.1109/WACV51458.2022.00249. (Retrieved: 2023-05-05).

[6] Ankan Kumar Bhunia et al. *Person Image Synthesis via Denoising Diffusion Model*. 2022. DOI: 10.48550/ARXIV.2211.12500. URL: https://arxiv.org/abs/2211.12500. (Retrieved: 2023-02-15).

[7] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2018. DOI: 10.48550/ARXIV.1809.11096. URL: https://arxiv.org/abs/1809.11096. (Retrieved: 2023-03-16).

[8] Jason Brownlee. *A Gentle Introduction to Generative Adversarial Networks (GANs)*. 2019-07-19. URL: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (visited on 04/26/2023).

[9] Jooyoung Choi et al. "ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models". In: *CoRR* abs/2108.02938 (2021). arXiv: 2108.02938. URL: https://arxiv.org/abs/2108.02938. (Retrieved: 2023-02-16).

[10] Yunjey Choi et al. "StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8789–8797. DOI: 10.1109/CVPR.2018.00916. (Retrieved: 2023-02-15).

[11] Florinel-Alin Croitoru et al. *Diffusion Models in Vision: A Survey*. 2022. DOI: 10.48550/ARXIV.2209.04747. URL: https://arxiv.org/abs/2209.04747. (Retrieved: 2023-02-06).

[12] Andreas Bach Daasbjerg and Damián Leporis. *Image synthesis for action localisation*. 2022. URL: https://projekter.aau.dk/projekter/da/studentthesis/billedsyntese-til-lokalisering-aktiviteter(cc080151-69a5-4e4f-933a-32d52e53d67d).html. (Retrieved: 2023-02-06).

[13] Andreas Bach Daasbjerg et al. *Image synthesis for action localisation*. Scientific paper on image synthesis for action localisation, not yet published. 2022. (Retrieved: 2023-02-06).

[14] DeepFindr. *Diffusion models from scratch in Pytorch*. July 18, 2022. URL: https://www.youtube.com/watch?v=a4Yfz2FxXiY&t=2s&ab_channel=DeepFindr (visited on 02/15/2023).

[15] Wan-Cyuan Fan et al. *Frido: Feature Pyramid Diffusion for Complex Scene Image Synthesis*. 2022. DOI: 10.48550/ARXIV.2208.13753. URL: https://arxiv.org/abs/2208.13753. (Retrieved: 2023-02-15).

[16] Kevin Feasel. *Finding ghosts in your data : anomaly detection techniques with examples in Python*. eng. New York, New York: Apress, 2022. ISBN: 9781484288702. (Retrieved: 2023-04-20).

[17] Philipp Fischer et al. *FlowNet: Learning Optical Flow with Convolutional Networks*. 2015. arXiv: 1504.06852 [cs.CV]. (Retrieved: 2023-04-27).

[18] Leon Gatys, Alexander Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: (Aug. 2015). DOI: 10.1167/16.12.326. (Retrieved: 2023-02-14).

[19] *General Data Protection Regulation (GDPR) - Official Legal Text*. (Retrieved: 2023-02-16).

[20] Allison Del Giorno, J. Andrew Bagnell, and Martial Hebert. "A Discriminative Framework for Anomaly Detection in Large Videos". In: *CoRR* abs/1609.08938 (2016). arXiv: 1609.08938. URL: http://arxiv.org/abs/1609.08938. (Retrieved: 2023-05-20).

[21] Xinyu Gong et al. "AutoGAN: Neural Architecture Search for Generative Adversarial Networks". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 3223–3233. DOI: 10.1109/ICCV.2019.00332. (Retrieved: 2023-05-05).

[22] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume*

2. NIPS'14. Montreal, Canada: MIT Press, 2014, 2672–2680. (Retrieved: 2023-05-04).

[23] Mahmudul Hasan et al. *Learning Temporal Regularity in Video Sequences*. 2016. arXiv: 1604.04574 [cs.CV]. (Retrieved: 2023-03-25).

[24] Wouter van Heeswijk. *Precision and Recall — A Comprehensive Guide With Practical Examples*. Jan. 31, 2022. URL: https://towardsdatascience.com/precision-and-recall-a-comprehensive-guide-with-practical-examples-71d614e3fc43 (visited on 03/16/2023).

[25] Sasaki Hiroshi. *Regarding UNIT-DDPM: UNpaired Image Translation with Denoising Diffusion Probabilistic Models*. 2023. Email correspondence, private communication.

[26] Jonathan Ho et al. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design". In: *CoRR* abs/1902.00275 (2019). arXiv: 1902.00275. URL: http://arxiv.org/abs/1902.00275. (Retrieved: 2023-03-17).

[27] Soonmin Hwang et al. "Multispectral Pedestrian Detection: Benchmark Dataset and Baselines". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015. (Retrieved: 2023-02-14).

[28] Radu Tudor Ionescu et al. *Unmasking the abnormal events in video*. 2017. arXiv: 1705.08182 [cs.CV]. (Retrieved: 2023-05-20).

[29] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *CoRR* abs/1611.07004 (2016). https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/blob/master/docs/tips.md. arXiv: 1611.07004. URL: http://arxiv.org/abs/1611.07004. (Retrieved: 2023-02-20).

[30] Anil K. Jain, Jianchang Mao, and K.M. Mohiuddin. "Artificial neural networks: a tutorial". In: *Computer* 29.3 (1996), pp. 31–44. DOI: 10.1109/2.485891. (Retrieved: 2023-04-15).

[31] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: http://arxiv.org/abs/1603.08155. (Retrieved: 2023-05-04).

[32] James M. Joyce. "Kullback-Leibler Divergence". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327. URL: https://doi.org/10.1007/978-3-642-04898-2_327. (Retrieved: 2023-04-10).

[33] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018. DOI: 10.48550/ARXIV.1812.04948. URL: https://arxiv.org/abs/1812.04948. (Retrieved: 2023-05-05).

[34] My Kieu et al. "Domain Adaptation for Privacy-Preserving Pedestrian Detection in Thermal Imagery". In: *Image Analysis and Processing – ICIAP 2019*.

Vol. 11752. Lecture Notes in Computer Science. Springer International Publishing, 2019, 203–213. ISBN: 978-3-030-30644-1. DOI: 10.1007/978-3-030-30645-8_19. URL: http://link.springer.com/10.1007/978-3-030-30645-8_19. (Retrieved: 2023-02-14).

[35] Vladimir V. Kniaz et al. "ThermalGAN: Multimodal Color-to-Thermal Image Translation for Person Re-Identification in Multispectral Dataset". In: *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, 2018. (Retrieved: 2023-02-14).

[36] Shu Kong and Deva Ramanan. "OpenGAN: Open-Set Recognition via Open Data Generation". In: *ICCV*. 2021. (Retrieved: 2023-03-01).

[37] Okan Köpüklü, Xiangyu Wei, and Gerhard Rigoll. "You Only Watch Once: A Unified CNN Architecture for Real-Time Spatiotemporal Action Localization". In: *CoRR* abs/1911.06644 (2019). arXiv: 1911.06644. URL: http://arxiv.org/abs/1911.06644. (Retrieved: 2023-02-20).

[38] Anders Krogh. "What are artificial neural networks?" In: *Nature biotechnology* 26.2 (2008), pp. 195–197. (Retrieved: 2023-04-05).

[39] Wen Liu et al. "Future Frame Prediction for Anomaly Detection - A New Baseline". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6536–6545. DOI: 10.1109/CVPR.2018.00684. (Retrieved: 2023-04-27).

[40] Cewu Lu, Jianping Shi, and Jiaya Jia. "Abnormal Event Detection at 150 FPS in Matlab". In: 2013. (Retrieved: 2023-05-14).

[41] Weixin Luo, Wen Liu, and Shenghua Gao. "A Revisit of Sparse Coding Based Anomaly Detection in Stacked RNN Framework". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 341–349. DOI: 10.1109/ICCV.2017.45. (Retrieved: 2023-05-20).

[42] Weixin Luo, Wen Liu, and Shenghua Gao. "Remembering history with convolutional LSTM for anomaly detection". In: *2017 IEEE International Conference on Multimedia and Expo (ICME)*. 2017, pp. 439–444. DOI: 10.1109/ICME.2017.8019325. (Retrieved: 2023-05-18).

[43] Tianxiang Ma et al. "MUST-GAN: Multi-level Statistics Transfer for Self-driven Person Image Generation". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 13617–13626. DOI: 10.1109/CVPR46437.2021.01341. (Retrieved: 2023-05-05).

[44] Vijay Mahadevan et al. "Anomaly detection in crowded scenes". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 1975–1981. DOI: 10.1109/CVPR.2010.5539872. (Retrieved: 2023-05-14).

[45] Xudong Mao et al. "Least Squares Generative Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2813–2821. DOI: 10.1109/ICCV.2017.304. (Retrieved: 2023-04-29).

[46] Michael Mathieu, Camille Couprie, and Yann LeCun. *Deep multi-scale video prediction beyond mean square error*. 2016. arXiv: 1511.05440 [cs.LG]. (Retrieved: 2023-04-27).

[47] Varun Menon and Kevin Stephen. "Re Learning Memory Guided Normality for Anomaly Detection". In: *ML Reproducibility Challenge 2020*. 2021. URL: https://openreview.net/forum?id=vvLWTXkJ2Zv. (Retrieved: 2023-03-27).

[48] Microsoft. *What is the Windows Subsystem for Linux?* 2022-08-12. URL: https://docs.microsoft.com/en-us/windows/wsl/about (visited on 05/26/2023).

[49] Sarang Narkhede. *Understanding AUC - ROC Curve*. 2018-06-26. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (visited on 05/10/2023).

[50] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved Denoising Diffusion Probabilistic Models". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8162–8171. URL: https://proceedings.mlr.press/v139/nichol21a.html. (Retrieved: 2023-02-06).

[51] Ivan Nikolov et al. "Seasons in Drift: A Long Term Thermal Imaging Dataset for Studying Concept Drift". In: (2021). URL: https://openreview.net/forum?id=LjjqegBNtPi. (Retrieved: 2023-02-16).

[52] OpenAI. *improved-diffusion*. Official github for Improved Denoising Diffusion Probabilistic Models. 2021. URL: https://github.com/openai/improved-diffusion (visited on 02/28/2023). (Retrieved: 2023-02-28).

[53] Debabrata Pal et al. "MORGAN: Meta-Learning-Based Few-Shot Open-Set Recognition via Generative Adversarial Network". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, pp. 6295–6304. (Retrieved: 2023-05-05).

[54] Cristina Palmero et al. "Multi-modal rgb–depth–thermal human body segmentation". In: *International Journal of Computer Vision* 118.2 (2016), pp. 217–239. (Retrieved: 2023-02-14).

[55] Hyunjong Park, Jongyoun Noh, and Bumsub Ham. "Learning Memory-guided Normality for Anomaly Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14372–14381. (Retrieved: 2023-03-19).

[56] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG]. (Retrieved: 2023-03-19).

[57] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *CoRR* abs/2112.10752 (2021). arXiv: 2112.10752. URL: https://arxiv.org/abs/2112.10752. (Retrieved: 2023-02-10).

[58] Mohammad Sabokrou et al. "Adversarially Learned One-Class Classifier for Novelty Detection". In: *CoRR* abs/1802.09088 (2018). arXiv: 1802.09088. URL: http://arxiv.org/abs/1802.09088. (Retrieved: 2023-03-16).

[59] Chitwan Saharia et al. "Palette: Image-to-Image Diffusion Models". In: *CoRR* abs/2111.05826 (2021). arXiv: 2111.05826. URL: https://arxiv.org/abs/2111.05826. (Retrieved: 2023-02-10).

[60] Hiroshi Sasaki, Chris G. Willcocks, and Toby P. Breckon. "UNIT-DDPM: UNpaired Image Translation with Denoising Diffusion Probabilistic Models". In: *CoRR* abs/2104.05358 (2021). arXiv: 2104.05358. URL: https://arxiv.org/abs/2104.05358. (Retrieved: 2023-02-06).

[61] Thomas Schlegl et al. *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. 2017. arXiv: 1703.05921 [cs.CV]. (Retrieved: 2023-05-21).

[62] Terrence J. Sejnowski and Charles R. Rosenberg. "NETtalk: a parallel network that learns to read aloud". In: 1988. (Retrieved: 2023-04-15).

[63] Jaskirat Singh, Stephen Gould, and Liang Zheng. *High-Fidelity Guided Image Synthesis with Latent Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2211.17084. URL: https://arxiv.org/abs/2211.17084. (Retrieved: 2023-02-10).

[64] Sorina Smeureanu et al. "Deep Appearance Features for Abnormal Behavior Detection in Video". In: *Image Analysis and Processing - ICIAP 2017*. Ed. by Sebastiano Battiato et al. Cham: Springer International Publishing, 2017, pp. 779–789. ISBN: 978-3-319-68548-9. (Retrieved: 2023-05-20).

[65] Angela A. Sodemann, Matthew P. Ross, and Brett J. Borghetti. "A Review of Anomaly Detection in Automated Surveillance". eng. In: *IEEE transactions on systems, man and cybernetics. Part C, Applications and reviews* 42.6 (2012), pp. 1257–1272. ISSN: 1094-6977. (Retrieved: 2023-04-20).

[66] Ryan Szeto et al. "HyperCon: Image-To-Video Model Transfer for Video-To-Video Translation Tasks". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 3079–3088. DOI: 10.1109/WACV48630.2021.00312. (Retrieved: 2023-05-05).

[67] Anwaar Ulhaq, Naveed Akhtar, and Ganna Pogrebna. *Efficient Diffusion Models for Vision: A Survey*. 2022. DOI: 10.48550/ARXIV.2210.09292. URL: https://arxiv.org/abs/2210.09292. (Retrieved: 2023-02-06).

[68] Ramya Vidiyala. *6 Types of Neural Networks Every Data Scientist Must Know*. 2020-17-2020. URL: https://towardsdatascience.com/6-types-of-neural-networks-every-data-scientist-must-know-9c0d920e7fce (visited on 04/26/2023).

[69] Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR) - A practical guide*. Springer International Publishing, 2017. DOI: https://doi-org.zorac.aub.aau.dk/10.1007/978-3-319-57959-7. (Retrieved: 2023-02-16).

[70] Muhammad Zaigham Zaheer et al. "Old is Gold: Redefining the Adversarially Learned One-Class Classifier Training Paradigm". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14183–14193. (Retrieved: 2023-03-15).

[71] Houssam Zenati et al. *Adversarially Learned Anomaly Detection*. 2018. arXiv: 1812.02288 [cs.LG]. (Retrieved: 2023-05-20).

[72] Houssam Zenati et al. *Efficient GAN-Based Anomaly Detection*. 2019. arXiv: 1802.06222 [cs.LG]. (Retrieved: 2023-05-21).

[73] Yu Zeng et al. *SceneComposer: Any-Level Semantic Image Synthesis*. 2022. DOI: 10.48550/ARXIV.2211.11742. URL: https://arxiv.org/abs/2211.11742. (Retrieved: 2023-02-11).

[74] Han Zhang et al. "StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5908–5916. DOI: 10.1109/ICCV.2017.629. (Retrieved: 2023-05-05).

[75] Lichao Zhang et al. "Synthetic data generation for end-to-end thermal infrared tracking". In: *CoRR* abs/1806.01013 (2018). arXiv: 1806.01013. URL: http://arxiv.org/abs/1806.01013. (Retrieved: 2023-02-20).

[76] Yingying Zhang et al. "Single-Image Crowd Counting via Multi-Column Convolutional Neural Network". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 589–597. DOI: 10.1109/CVPR.2016.70. (Retrieved: 2023-05-14).

[77] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *CoRR* abs/1703.10593 (2017). arXiv: 1703.10593. URL: http://arxiv.org/abs/1703.10593. (Retrieved: 2023-02-20).

A Resources

A.1 Github repository

A github repository has been created to host the code used in this thesis. The repository can be found at: `https://github.com/TheCanMoun10/VGIS10_GAN`

B Prior report

This chapter contains a literature review regarding recent developments in the usage of diffusion models in image-to-image translation (I2I) and style transfer methods. Recent work in the field of applying diffusion models to I2I tasks will be reviewed, as these will give insight into how a diffusion model can be developed to do the task of RGB to Thermal style transfer.

Thermal cameras are a reliable choice under low light conditions, in places where occlusions occur regularly, and during night time. An added benefit of using and working with thermal images is its privacy-preserving characteristics [34]. which is of great importance in the context of compliance with video surveillance with respect to the General Data Protection Regulation (GDPR) in the European Union [19, 69]. Having thermal activity recognition datasets could also help in detecting activities in low-light and nighttime settings. There is however a limited amount of datasets which does have a thermal counterpart - which is also the case for activity recognition datasets. For this reason, style transfer from the RGB domain to the thermal domain is a much-needed method, as it allows for the creation of potential multi-modal datasets from a single dataset. In computer vision, the task of transferring one possible representation of a scene into another, such as RGB imagery to thermal imagery can be described by the terms style transfer" or "image-to-image translation". By traditionally using CNNs, style transfer allows for mathematically accurate definitions of the contents and styles of images [18]. Having both content and style of the images, loss functions, which describe the difference between the style and content of two images, can be defined [18]. Applying backpropagation the the pixels of one image may be updated to more closely resemble the style or content of the other image [18]. The ThermalGAN framework [35] have provided promising results in style transfer from RGB to thermal domain images, for use in person re-identification. Through the use of metadata, Kniaz *et al*. can produce realistic and diverse images that show small temperature contrasts [35]. However, more often metadata is not readily available for the style transfer task.

Research shows that including information from different modalities improves the performance of deep neural networks, but existing datasets are often lacking

Figure B.1: Results on indoor images using images from the VAP dataset [54]. The second row is ground truth images. The bottom row is the synthetic images created from the model made in [13, 12].

in terms of the number of modalities. It would therefore be preferable to perform a translation task from RGB to the thermal domain, which helps to generate the thermal counterpart of the existing RGB datasets. Some of the initial efforts, from a previous semester project, successfully translated the RGB to the thermal domain with the help of generative adversarial network (GAN) based methods [13] [12]. However, the GAN-based methods can suffer from "mode collapse", where they fail to cover the entire distribution. Results from the semester project showed that the approaches work well for the translation of indoor scenes in Figure B.1. However, the method fails to work for outdoor scenes shown in Figure B.2.

Figure B.2: Results on outdoor images. The image to the left is an outdoor image from the KAIST-Multispectral Pedestrian dataset [27]. The image to the right is a synthetic outdoor image created by the model in [13, 12]

A new class of models emerged recently that resolved this issue, these models are known as "diffusion models" [50]. The goal of this project is to explore diffusion models for translating RGB images to the thermal domain.

B.1 Prior work

B.1.1 Image synthesis for action localisation

The main purpose of this semester is to investigate how a computer vision system could be designed to localise actions in a large thermal dataset [12]. With the project a three-step proof of concept pipeline is introduced capable of (1) applying domain transfer models in the form of GANs (Pix2Pix [29] and CycleGAN [77]) to generate synthetic thermal action localisation dataset, (2) training and finetuning an action localisation network with the new thermal dataset and (3) providing annotations for any actions localised in large thermal dataset [51] by utilising the thermal action localisation network [12]. For the action localisation, the pipeline utilises the You Only Watch Once (YOWO) framework [37]. The proposed pipeline can be seen in Figure B.3. The focus of the report was limited to working on creating a style transfer model, and since Pix2Pix [29] and CycleGAN [77] were utilised both a supervised and unsupervised style transfer model was created. For the supervised approach, the model was trained on 4162 RGB-Thermal image pairs from the KAIST-dataset [27], and parameters were trained on 1617 images from the same dataset [12]. The unsupervised approach was trained on a dataset containing 4006 images in total from both the RGB (2003) and Thermal (2003) domains.

Results from testing how different parameters affected the resulting synthetic image were less than ideal. It was shown that the supervised approach using Pix2Pix was able to synthetically generate images that were close to converging into the thermal domain [12]. Whereas the unsupervised approach only was able to generate images that looked like grayscale images. Figure B.4 shows the resulting comparison study of the models with the ground truth thermal images from the KAIST dataset.

B.1.2 Image Synthesis: RGB to Thermal Domain

Extending on the work from the semester report a scientific paper was also created [1] where the focus is to introduce a GAN pipeline which can do style-transfer from the RGB domain to the thermal domain in both a supervised and unsupervised approach, without the need for metadata using well known GAN approaches (Pix2Pix [29] and CycleGAN [77]), to address the scarcity of paired thermal datasets [13]. The generator's capability to produce thermal images was tested by applying different data augmentations from torch-vision to the dataset. In total three augmentations were tested and applied to the VAP dataset [54] in order: Autocontrast, Adjusting sharpness of images, and applying Gaussian blur. Both models use most of the original implementations from [29] and [77], with minor changes to the data

[1] Has yet to be published, still a work in progress.

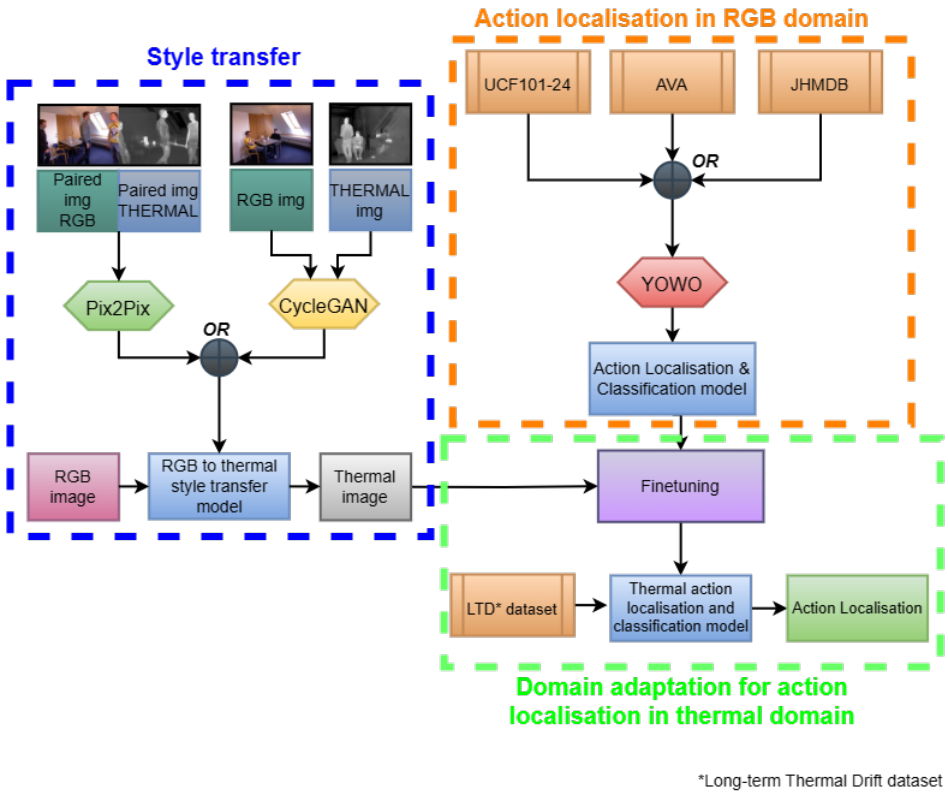


Figure B.3: The proposed system from the *Image synsthesis for action localisation* report [12]. Step 1: Style transfer, depending on which architecture is used either paired images or unpaired images from RGB and Thermal domain are used to generate the Style transfer model. Step 2: Action localisation in RGB domain, the YOWO framework is trained on one of three action localisation datasets to generate an action localisation and classification model. Step 3: The model from step 1 is used to fine-tune the model from step 2 to generate a thermal action localisation and classification model, which is used to annotate the Long-term Thermal Drift dataset [51].

loader of each model. The models were evaluated through ablation to decide on the best model parameters. After determining the best model parameters for the supervised approach, it was trained for 20 epochs and its ability to generate synthetic images was tested after 5, 10, 15 and 20 epochs. This led to the conclusion that the model was still learning even after 20 epochs, and it was as such trained for 50 epochs to see how the model performed the translation, the result of this test can be seen in the bottom row of Figure B.1. The unsupervised approach was tested on how well the different epoch steps translate to the thermal domain, due to it being computationally heavy (two generators and discriminators are trained). Like in [12] the unsupervised approach provided less than stellar results and only the supervised approach was evaluated for a final test. To assess the quality and

Figure B.4: Comparison study from the *Image synsthesis for action localisation* report. The top row shows the RGB images given as input to the models, the second row shows the ground truth thermal images, the third row the thermal images generated by the supervised model, fourth row the thermal images created by the unsupervised model [13].

performance of the model a survey with 30 participants was conducted, in which they were asked how well a randomly chosen synthetic image resembled the respective ground truth thermal image (sample images can be seen in the bottom row of Figure B.1). 34% of respondents thought the synthetic images had a 90% resemblance to the ground truth images, 32% thought the synthetic image had a 75% resemblance to the ground truth images, and 13% thought the synthetic images had a 100% resemblance to the ground truth images. The survey results can be seen in figure B.6

As the paper has yet to be published (as of the writing of this thesis) the contribution the paper will serve is a style transfer model that is capable of creating synthetic thermal images from any set of RGB images. At its current stage, however, the model has some limitations in its performance on outdoor images. It is not able to properly translate outdoor images to the thermal domain as there is noise the model is not able to account for (examples of this can be seen in Figure B.2), and it lacks some form of novelty to distinguish it from other domain-transfer approaches such as [35] and [75]. Thus it is the hope that with this thesis, the novelty of the paper can be introduced.

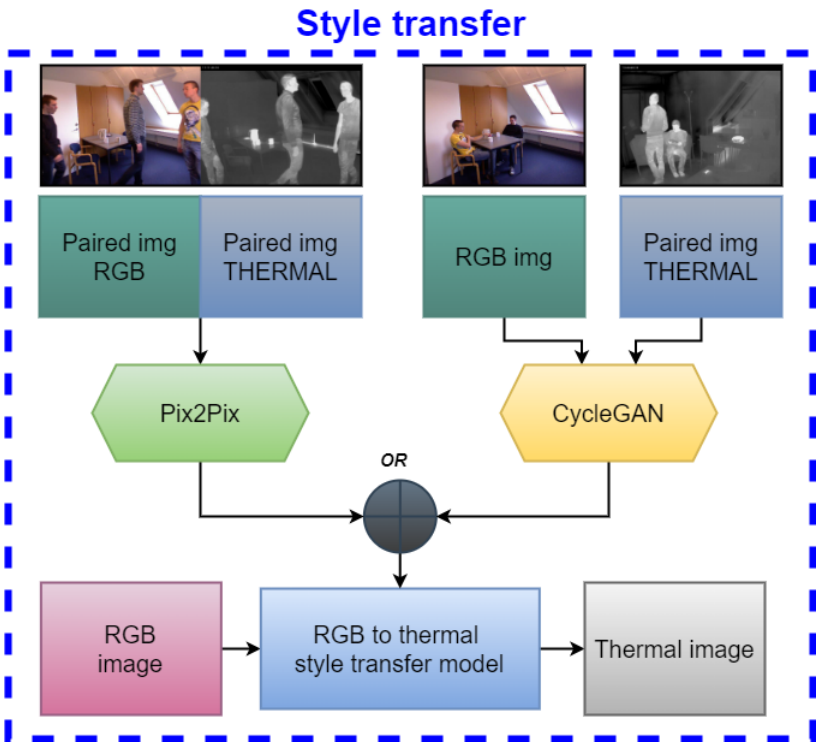


Figure B.5: The proposed system from *Image synsthesis: RGB to thermal domain* [13].

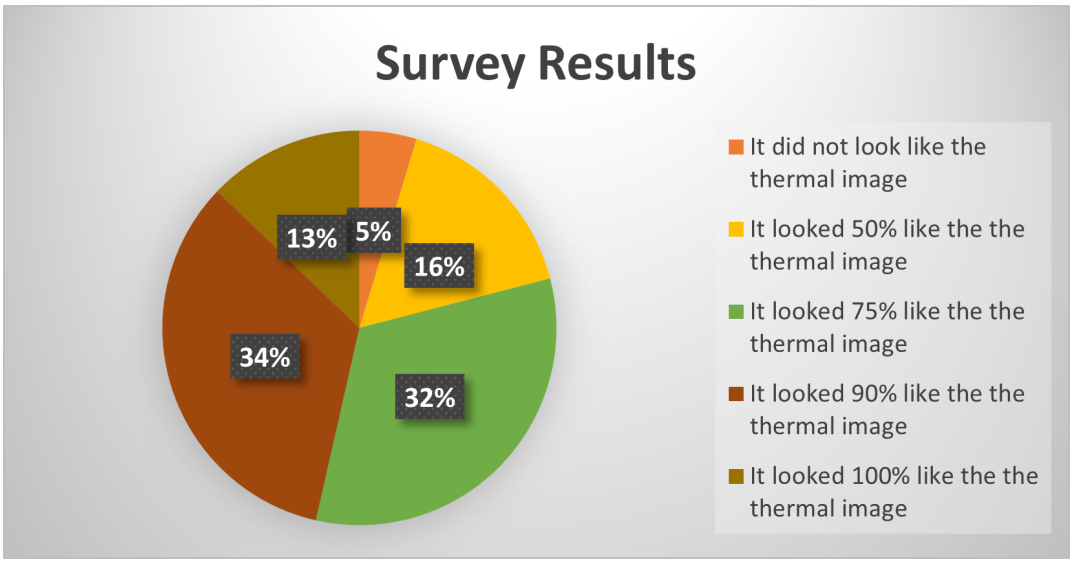


Figure B.6: Survey results of the final model in [13]. A combined 79% thought the synthetic images had a resemblance of 75% and above.

C Diffusion Models

C.1 Recent developments in image synthesis

In recent years diffusion models have seen a rise in usage across the field of image synthesis. Ranging from anywhere between Person Image Synthesis [6], Unpaired Image translation [60] to High-resolution and High-fidelity image synthesis (super-resolution) [57, 63] as well as other areas [59, 73, 15, 9]. The reason Diffusion models have become a topic of large interest in computer vision is due to their generative capabilities. Diffusion models can generate images with a great level of detail and are also able to generate many diverse generated examples [11]. A diffusion model's ability to learn the latent representation of an image or image domain has also shown to be useful when applied to discriminative tasks such as anomaly detection and classification [11].

C.1.1 Diffusion model framework

Diffusion models can be seen as a class of probabilistic generative models learning to reverse a process that has gradually destroyed the training data (usually in the form of gradually adding noise to the training data) [11]. This is what the original idea of the models stems from modelling a specific distribution from random noise [67]. Meaning that the distribution of the generated samples should be as close to the original sample as possible [67]. Common for all diffusion models is that they share the same baseline processes: Foward and Backward diffusion [67, 11, 14], see Figure C.1.

In the forward process, the noise is added to the images, done through a Markov decision process. The noise added usually only depends on the previous image in the training dataset, and it is sampled using a conditional Gaussian distribution with a mean that depends on the previous image and a fixed variance [14]. The noise added to the images is specified by a variance schedule, that describes how much noise is added to the images at a specific time step [14, 67]. The reverse diffusion process called reverse diffusion, is where the strength of the diffusion

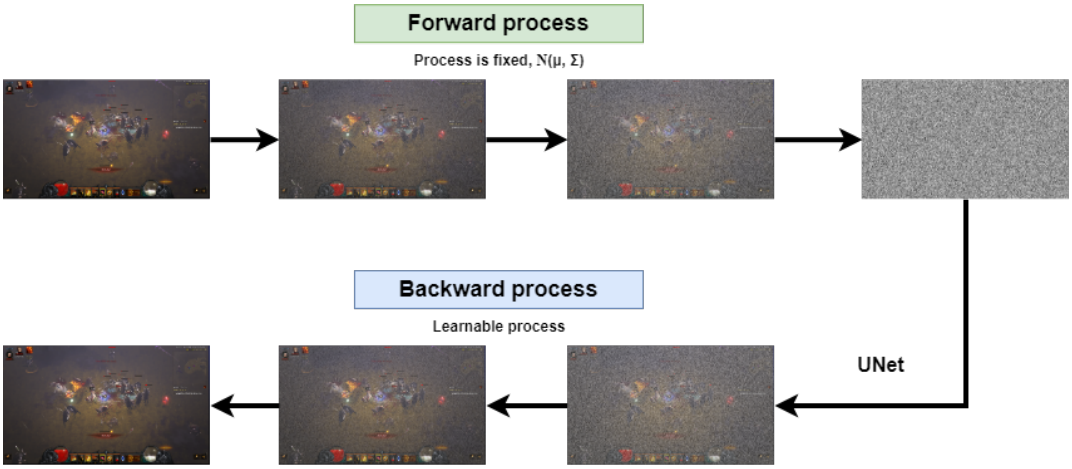


Figure C.1: Forward and backward process of a diffusion model. During the forward process noise (often Gaussian noise [14]) is added to the input image at each time step gradually destroying it, using a noise scheduler. The forward process is fixed. The backward process is a learnable process often utilizing UNets [14, 50]. The transition from one latent to another in the latent space is learned at random time steps. Doing sampling all time steps are sampled to reconstruct the input image.

models are - since the goal of the model is to learn the reverse process [67]. The reverse process can be done by training a neural network (usually done using U-Nets [14]) to approximate the probabilities such that the diffusion can be reversed [67]. The backward process starts at a given time-step with Gaussian noise with zero mean and unit variance [14]. Given the current time-step, the transition from one latent to the next latent in the latent space is predicted, making the model learn the probability density of an earlier time-step [14]. During training the timesteps in the process are randomly sampled, such that it does not go through the entire sequence of noisy images. At sampling, however, all timesteps are sampled, as the process has gone from pure noise to a final image [14].

For this thesis, two papers detailing diffusion models were found relevant for the project to get a baseline diffusion model working.

C.1.2 Improved Denoising Diffusion Probabilistic Models

In [50], a paper by OpenAI, Nichol and Dhariwal show that with simple modifications denoising diffusion probabilistic models (DDPMs) can achieve competitive loglikelihoods and still maintain high sample quality when compared to GANs. Additionally, this comparison precision and recall is used to compare how well a DDPM covers the target distribution to a GAN [50]. To study the effects of different modifications to a DDPM network, Nichol and Dhariwal train fixed model architectures with fixed hyperparameters on the CIFAR-10 (which has seen its fair usages for DDPMs) and ImageNet 64×64 datasets [50]. In the context of DDPMs the Im-

ageNet 64×64 dataset provides a good trade-off between diversity and resolution - allowing a model to be trained quickly and without the worry of overfitting. ImageNet 64×64 has also been largely used in generative modelling, which allows for a comparison between DDMPs and GAN models [50]. Nichol and Dhariwal suggest improvements for some aspects of DDPMs: the log-likelihood and the noise schedule. For the log-likelihood Nichol and Dhariwal take their baseline in the work of Ho *et al*.[26], and found they could achieve an initial boost in the log-likelihood by increasing the number of diffusion steps from 1000 to 4000 [50]. In the original implementation by Ho *et al*. [26] the variance in the Gaussian distribution is fixed and not learned by the network. To improve the log-likelihood Nichol and Dhariwal suggest that variance should be learned. They suggest parameterizing the variance as an interpolation between an upper and lower bound β_t and $\tilde{\beta}_t$ in the log domain. In improving the noise scheduler, Nichol and Dhariwal found that a linear noise schedule (like the one used in [26]) works well for high-resolution images, although it works sup-optimal for 64×64 and 32×32 images. Particularly they found that the end of the forward noising process is too noisy, which does not contribute much to the sample quality. The linear schedule falls towards zero fast and thus destroys information faster than necessary [50]. Instead, a cosine noise scheduler is proposed which has a linear drop-off in the middle of the diffusion process, and near the extremes (the beginning and the end) changes very little in the noise. For these two changes, Nichol and Dhariwal achieved competitive log-likelihood results with some of the best convolutional models. Their model is however worse when compared to fully transformer-based models [50].

GAN Comparison

For more coverage comparing the likelihoods of other DDPMs is a good proxy, however, it becomes difficult to compare GANs when only this metric is used. Nichol and Dhariwal instead use precision[1] and recall[2] when comparing with GANs.

Injecting class information through the same pathway as the timestep, in the form of embeddings, Nichol and Dhariwal allow their models to be class-conditional. They compare their DDPM model performance with the performance of the BigGAN-deep model [7], by training a small and large model. They found that BigGAN-deep outperforms their smaller model in terms of the Frechet Inception Distance (FID), while it struggles in terms of recall [50]. The samples generated in their testing can be seen in Figure C.2, where there is a high diversity in each of the classes

[1] A metric which penalizes false positives. Models with high precision are cautious in the process of labelling an element as positive. [24]

[2] A metric which penalize false negatives. Models with high recall go toward positive classification when in doubt.[24]

Figure C.2: Class conditional sample images sampled from ImageNet. Classes include ostrich, goldfinch, flamingo, redshank, pekinese, papillo, drake and spotted salamander. [50]

which suggests good coverage of the target distribution. With this comparison, Nichol and Dhariwal show that diffusion models are better at covering different modes of distribution than some comparable GANs [50].

Model training

The models trained by Nichol and Dhariwal were trained with 4000 diffusion steps and accordingly producing a single sample takes several minutes on modern GPUs [50]. For ImageNet they trained the models for 500K and 1500K training iterations, and for CIFAR-10 the models are trained for fewer iterations 200K and 500K. During model training, they found that having fixed variances their models suffer much more in sample quality compared to learnt variances when using a reduced number of sample steps. For the sampling of their models, they apply 100 steps to achieve near-perfect FID scores for their fully trained models. [50]

C.1.3 UNIT-DDPM: Unpaired Image Translation with Denoising Diffusion Probabilistic Models

In [60] Sasaki *et al*. propose an unpaired image-to-image translation method using DDPMs that requires no adversarial training. Traditionally for image-to-image translation tasks, a style transfer DNN that preserves the semantic content is used, these are also used in the randomization of image styles. Instead of using adversarial networks as their backend, Sasaki *et al*. propose to use a DDPM to mitigate the

limitation of unstable training and improve the quality of generated images [60]. Applying the latent information approximated by DDPM the UNIT-DDPM learns different domains of images and connects between the latent of the domains. Additionally, as a result, UNIT-DDPM allows for gradual sampling from noise (which is progressively denoised to images) with the target domain in a way that is related to the input source domain images.

The aim of [60] is to develop image-to-image translation between different images whose distribution is formed by a joint probability. Meaning the model, through empirical risk minimisation, learns the parameters of the models from a given data set of the source domain and target domains and subsequently infers the target domain images from the source domain images [60]. Usually in the inference of image translation, domain translation functions are used to translate the input images from the source to the target domain. By utilizing DDPMs in image-to-image translation these domain translation functions are no longer needed. The target images are synthesised, in a progressive manner, from the noisy domain images and the Gaussian noise, and during sampling the generative process is conditioned on the input source domain images perturbed by the forward diffusion process [60].

UNIT-DDPM evaluation

Sasaki *et al*. evaluate their model against prior unpaired image-to-image translation methods on publicly available datasets, where the ground truth input-output image pairs are available [60]. One such dataset is the KAIST Multispectral Pedestrian Dataset [27].

For the baselines the inferred output images from UNIT-DDPM are compared both quantitatively and qualitatively to CycleGAN, UNIT, MUNIT and DRIT++ [60]. For the datasets, all images are resized to 64×64 before initializing training.

The denoising models of UNIT-DDPM are implemented using U-Net (based on PixelCNN and Wide Resnet). They use the same sinusoidal position embedding as Ho et al [26] to encode the timestep and linear schedule, but the Swith unit is replaced with a Rectified Linear Unit (ReLU) and the group normalization is replaced by batch normalization. To reduce the computation time the self-attention block is removed. For the domain translation functions utilize ResNet architectures with the same depth of layers as the U-Net. During training the pair of training samples and fake domain samples are concatenated as the input. The model was trained with a batch size of 16 and 20,000 epochs using an ADAM optimizer. Figure C.3 shows a comparison between UNIT-DDPM and the baselines it is compared against. It can be seen that UNIT-DDPM qualitatively generates more realistic images. Sasaki *et al*. also found that UNIT-DDPM also did not suffer from mode

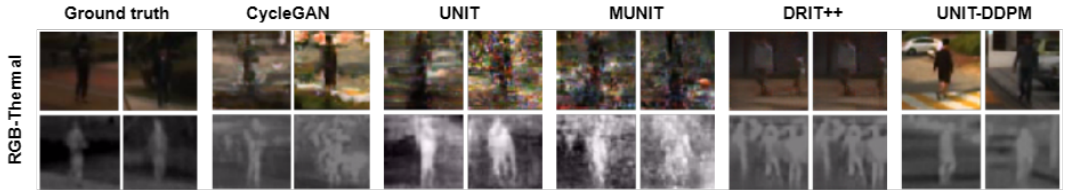


Figure C.3: Comparison of UNIT-DDPM with ground truth, CycleGAN, UNIT, MUNIT and DRIT++. Visually adapted from [60]

collapse and model training was more stable due to not having adversarial training [60].

Although UNIT-DDPM shows promising results there are certain limitations to the model. The model fails to learn global information of images when the images are of a higher resolution of the images are more than 64×64. Sasaki *et al*. found this limitation when they tested images with a resolution of 256×256 with the same network architecture and learning parameters as the lower-resolution images. Another drawback of the model is the inference time for image generation of DDPM. This drawback is stated to be solvable by either modifying the Markovian process such that the model implicitly models or by reducing the timesteps by using a learnable variance (like suggested in [50]) [60].

C.1.4 Remarks on diffusion model usage

Initially, it was planned to get a baseline of a diffusion model running such that a diffusion model (DM) framework for RGB to thermal style transfer could be created. However, upon researching and testing the topic it has become apparent that getting a baseline DM running is harder than initially thought and the time it takes to train a model is longer than thought. Ideally, the baseline for this report would be to run UNIT-DDPM [60] and test its different capabilities in the style transfer task. Here the trouble arose in the source code not being readily available for usage, as the authors are still working on fixing the problem of high-resolution images (larger than 64×64), additionally, the time taken to train a model was about a week on a 10GB memory GPU computer [25].

As an alternative baseline, getting the diffusion model from [50] running was tried, as the source code for this specific model is publicly available [52]. Getting the model to run on a cloud server with an available NVIDIA A40 GPU was relatively feasible. A model was initialised to train on the CIFAR-10 dataset for 1000 diffusion steps, however, training got stuck after two days of running the training on step 780, and the training was aborted after four days. This either implies a fault in the code or how the training was set up made the training get stuck. As for sampling the process took a bit shorter around one day to sample 10.000 images.

In total, it took five days to run a full run of the diffusion model. Although it was not a successful training it still gave an insight into how long training a diffusion model from scratch can take even on a dataset containing low-resolution images (32×32), with basic parameters. Depending on the amount of testing to be done in the experimental phase of the project, it could potentially take more than one week to finish one test, taking anywhere between three to seven weeks of testing models and that is assuming the first initial test would be yielding a good result. With this assumption, it has been deemed that is not feasible to be able to finish a model and the report on time, thus the project will be going in a different direction.

Instead of focusing on diffusion models, the project will instead focus on anomaly detection using outlier exposure and fake outlier generation-based GANs, by the use of OpenGANs. This change of topic still relates the project to style transfer and has been deemed more manageable in the given time frame for the project.

D Qualitative Results

D.1 Qualitative results of the proposed system

D.1.1 Video 3

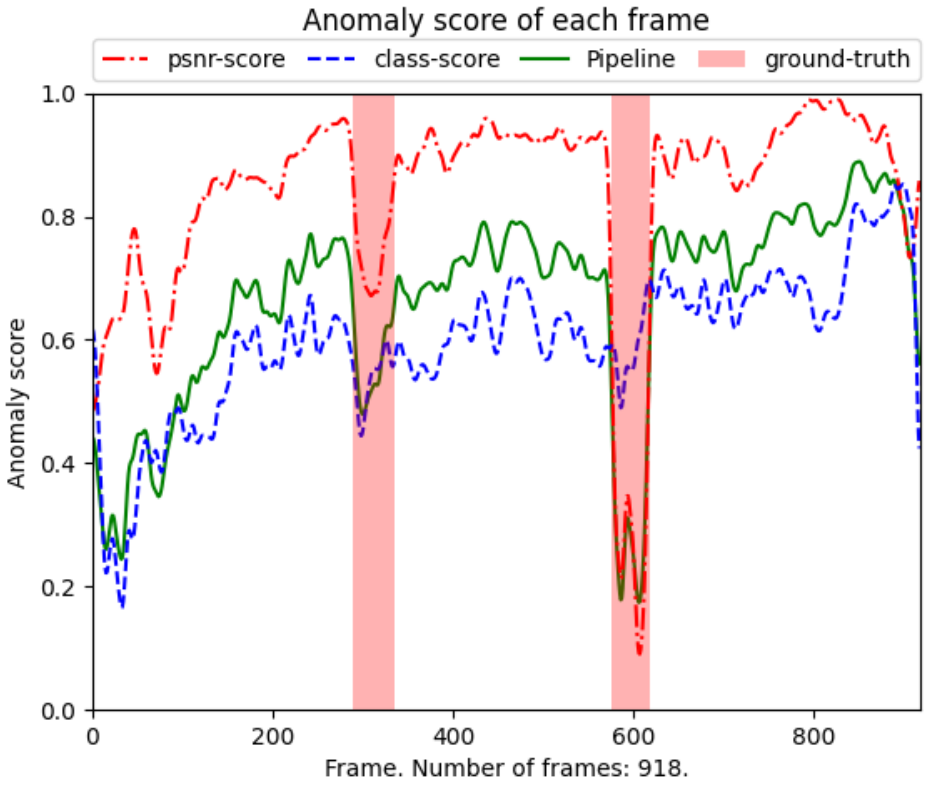


Figure D.1: Evaluation of the pipeline on video 3. The ground truth abnormal frames are marked with red boxes.

D.1.2 Video 4

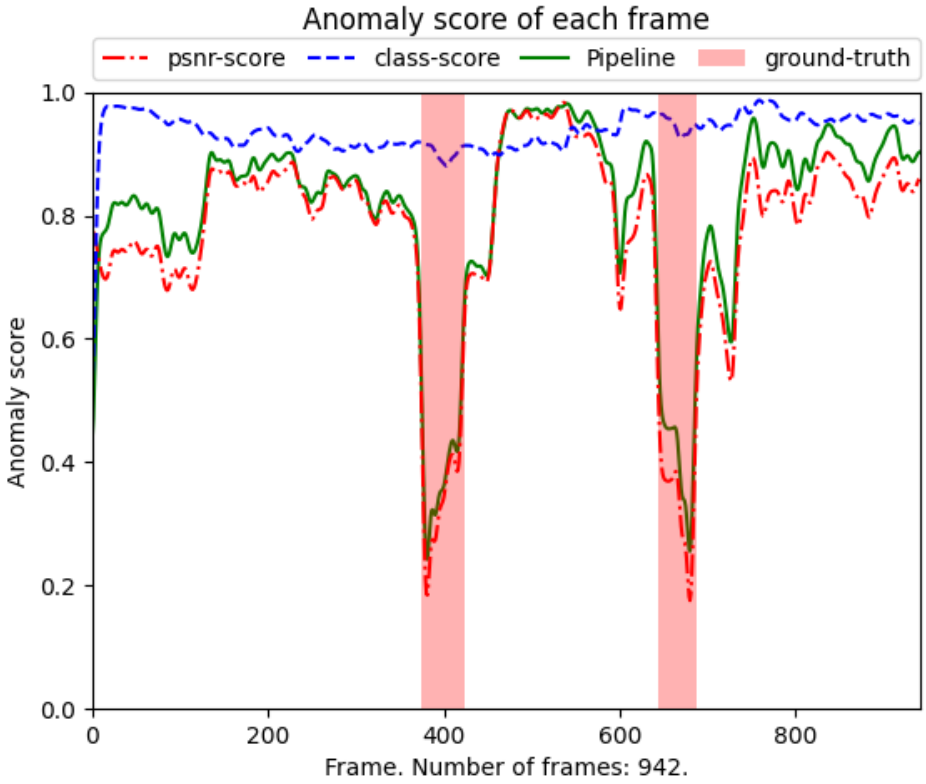


Figure D.2: Evaluation of the pipeline on video 4. The ground truth abnormal frames are marked with red boxes.

D.1.3 Video 6

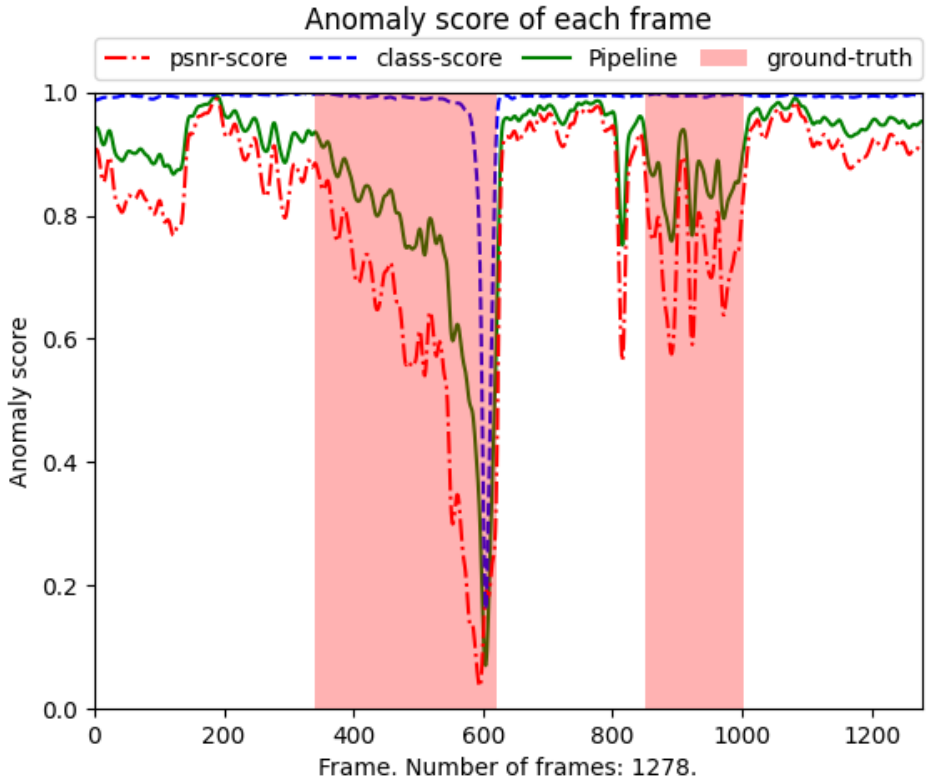


Figure D.3: Evaluation of the pipeline on video 6. The ground truth abnormal frames are marked with red boxes.

D.1.4 Video 17

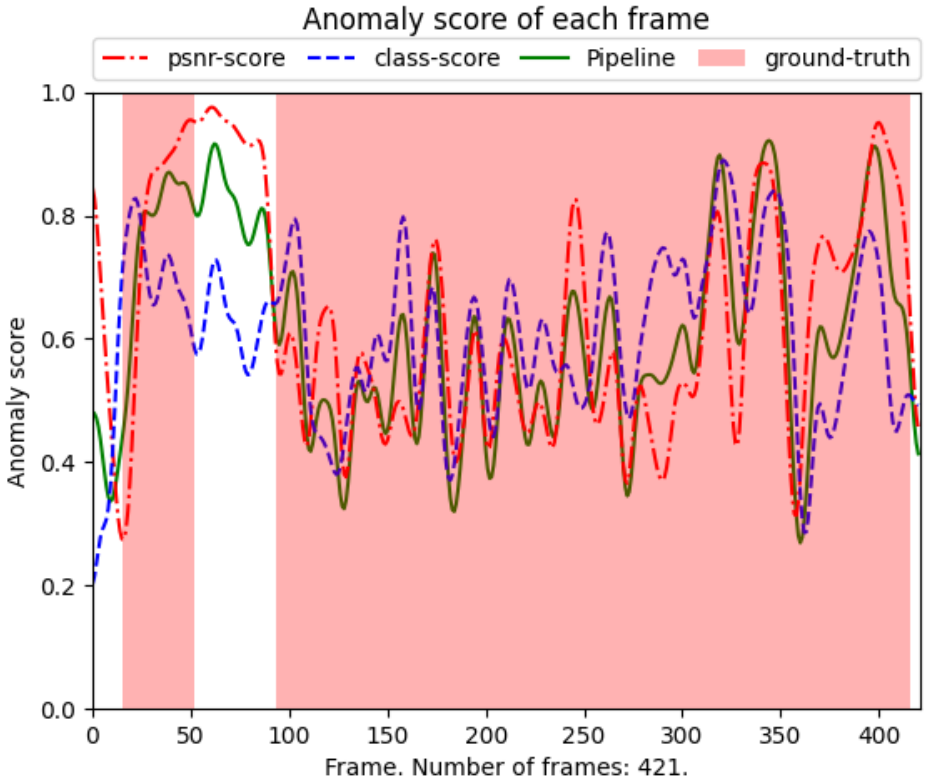


Figure D.4: Evaluation of the pipeline on video 17. The ground truth abnormal frames are marked with red boxes.

D.1.5 Video 18

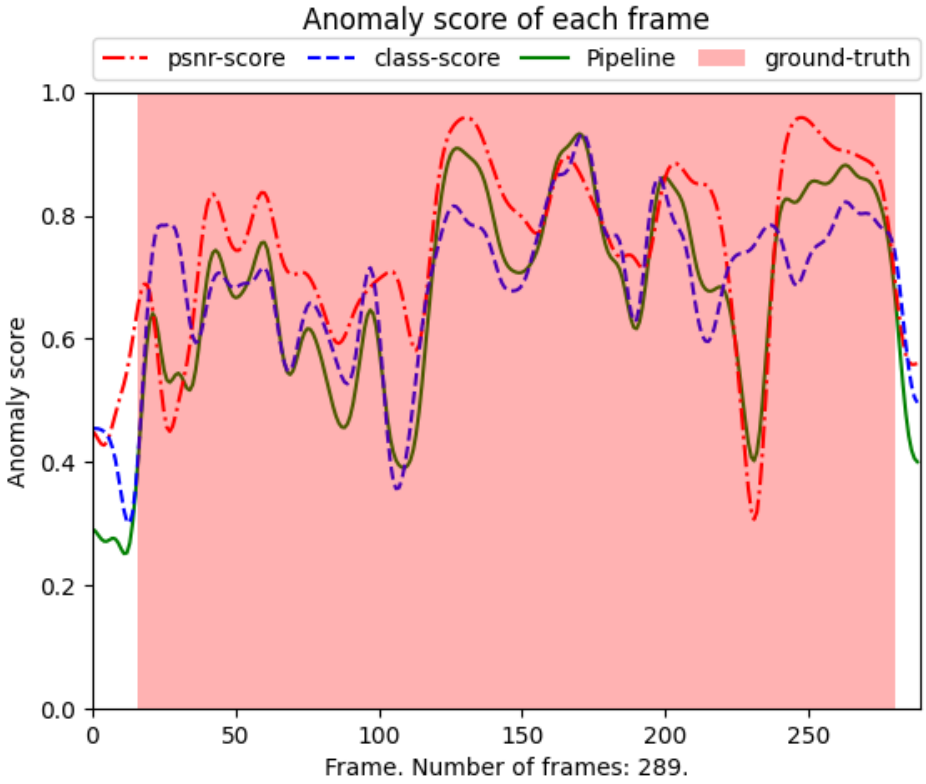


Figure D.5: Evaluation of the pipeline on video 18. The ground truth abnormal frames are marked with red boxes.

D.1.6 Video 21

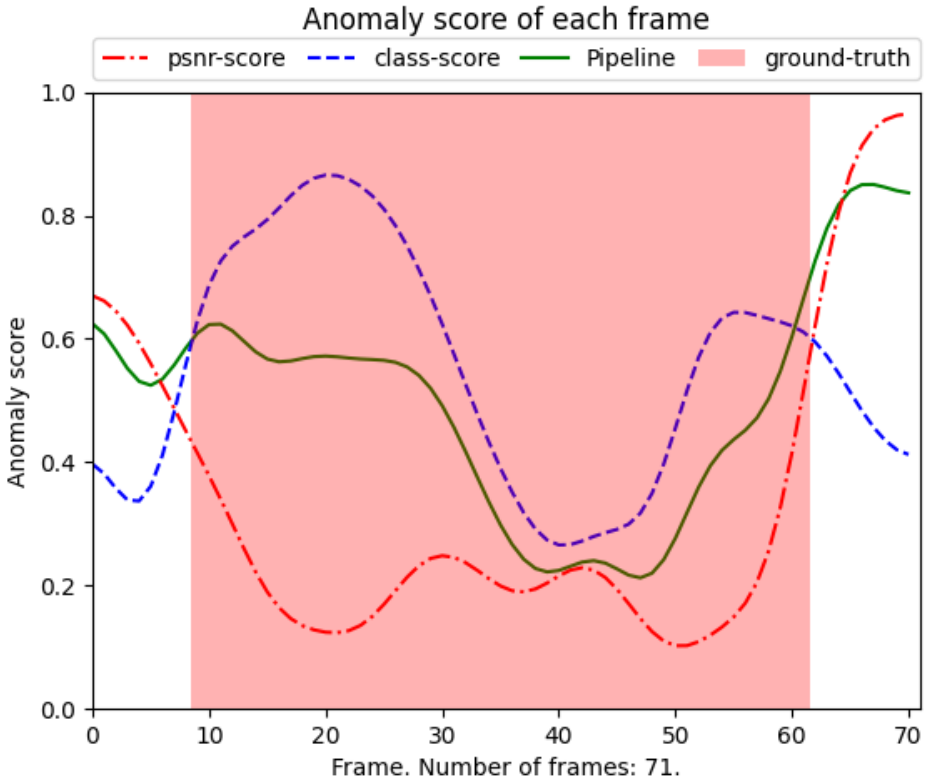


Figure D.6: Evaluation of the pipeline on video 21. The ground truth abnormal frames are marked with red boxes.