UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Anomaly detection with sinusoidal representation network

Author:
Yurii YELISIEIEV

Supervisor:
PhD Taras FIRMAN

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Yurii YELISIEIEV, declare that this thesis titled, "Anomaly detection with sinusoidal representation network" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Quality means doing it right when no one is looking."

Henry Ford

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Anomaly detection with sinusoidal representation network

by Yurii YELISIEIEV

Abstract

Anomaly detection in images helps to identify abnormal or unusual patterns in images relative to normal data. This issue is crucial in various domains, such as detecting abnormal areas in medical imaging, surface inspection, or photo editing. Many of the proposed methods require additional markup on the images. One way to solve this problem is image reconstruction. This method does not require additional markup on training data and trains on normal images. In this study, we present an approach that solves the Image Reconstruction problem by exploiting the Sinusoidal Representation Network (SIREN). SIREN is capable of modeling complex signals with great detail in small regions. We experimentally combined different loss functions for our architecture to improve the visual perception of the image. This study will also describe the different approaches we have tried in image reconstruction and our method's evolution.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Taras Firman for guiding me throughout the research process, providing answers to my questions late in the evening, and providing me with interesting ideas. I want to express my gratitude to the ELEKS DSO team for the big support and the opportunity to work on an interesting project. Finally, I want to thank all the teachers and my classmates with whom I managed to work during my studies for help and interesting discussions.

Contents

List of Figures

List of Tables

List of Abbreviations

SIREN	**Si**nusoidal **R**epres**e**ntation **N**etwork
SVM	**S**upport **V**ector **M**achine
RELU	**Re**ctified **L**inear **U**nit
MLP	**M**ulti**L**ayer **P**erceptron
GAN	**G**enerative **A**dversarial **N**etwork
WGAN	**W**asserstein **G**enerative **A**dversarial **N**etwork
DCGAN	Deep **Convolutional** **G**enerative **A**dversarial **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
FC	**F**ully **C**onnected
PRELU	**P**arametric **Re**ctified **L**inear **U**nit
CNN	**C**onvolution **N**eural **N**etwork

To my family

Chapter 1

Introduction

The development of computing machines and algorithms over the past 50 years has passed the stage of a new digital revolution. This has affected almost all parts of our lives and several industries - healthcare, manufacturing, finance, etc. Meanwhile, the field of digital image processing has grown exponentially in recent years. Tasks of digital image processing include classification, feature extraction, image compression, image restoration, and anomaly detection on images.

The popularity of automatic quality control in various domains is snowballing because automation will lead to faster and more accurate quality control. Automated systems can work without interruption providing faster and more robust control. The control with such systems is most often done by inspecting the visual appearance of an object. A search of the defects in images among ordinary translates to finding anomalous patterns that are not appropriate for the inspected object. Defect detection on the images could be helpful in various fields like fraud detection, health care, cybersecurity, etc. There is a long history of using different algorithms for finding anomalous samples. Classic approaches have a desirable performance on low-dimensional data, but images are high-dimensional, and algorithms are suffering under the curse of dimensionality. When there is a vast amount of data and is sparse, more training is needed to solve a problem. Therefore we need a more appropriate way for image analysis. Since Convolutional Neural Networks can work with high-dimensional data, they became a state-of-the-art method for image classification, segmentation, object detection, anomaly detection, etc. Neural networks are capable of modeling complex distributions of data. Therefore they apply to a wide range of problems such as anomaly detection and image generation. Neural networks could learn high-quality features at different levels that are extracted from images. Then these features are used to train a binary classification algorithm for anomaly detection. Instead of using a classification algorithm that is supervised in its manner, we could use these features to train unsupervised method like One-Class SVM[39]. This problem could also be solved by the generative model approach, like deep convolutional generative adversarial networks that work well for anomaly detection problems.

Sitzman et al. proposed a new method for signal representation parametrized by neural networks that could be used for image generation and further fore anomaly detection[41]. They showed that periodic activation functions are well suited for representing complex natural signals and their derivatives. These representations are capable of modeling signals with great detail. Implicit neural representation combined with other image anomaly detection methods could be potentially helpful for anomaly detection on images.

In Chapter 2, we provide the overview of the methods which were used during our study and anomaly detection approaches. In Chapter 3, we define the problem of anomaly detection and present an overview of the future work. In Chapter 4, we describe our SIREN based approach to anomaly detection with discriminator for anomaly score calculation. In Chapter 5, we describe experiments which were conducted, providing visual examples of the reconstruction on different dataset and details about implementation. Chapter 6 shows the results of best model reconstruction on damaged images and as anomaly scores. Finally in Chapter 7 we summarize the results and provide possible next steps.

Chapter 2

Related works

2.1 Implicit Neural Representations with Periodic Activation Functions

Signal representation is predominantly the way we look at an image. We can consider a signal as a function of several variables or a vector in a particular space. Different representations of the signal help to identify its additional properties. Sitzmann and others[41] suggested a brand-new method for representing any natural signal. Classic signal representation techniques are primarily discrete. For example, pictures are represented as a grid of pixels. Their proposed method allows representing the signal as a continuous function, for example, mapping the pixel coordinates to the RGB values. There is no such function that would parameterize a raw image as a mathematical formula. However, they tried to approximate this function with a neural network.

In general, they described a class of functions Φ that maps x to the particular value while satisfying the constraint F where Φ is parametrized with a neural network.

$$F(x, \Phi, \nabla_x \Phi, \nabla_x^2 \Phi, \ldots) = 0, \quad \Phi : x \to \Phi(x)$$

Constraint F specifies the relationship between the domain of the signal and the value of interest. These are the coordinates of the pixel on the grid and its intensity value in RGB or grayscale in the images. The use of continuous representations of the signal has several notable advantages over discrete representations. The fact that the function is defined on a continuous interval of x allows representing the picture with great detail without reference to the size of the grid in discrete representation. However, this ability is limited by the neural network's capacity used for the parametrization of the function. Such representation is not only continuous but also differentiable. That implies that higher-order gradients and derivatives can be calculated analytically, which is beneficial in various applications.

Some previous studies have tried to use Rectified Linear Unit (ReLU) based multilayer perceptrons (MLPs) for implicit neural representations. These studies have shown that models with ReLU do not have enough capacity to model the signal in great detail and its derivatives. Such results could be explained by the fact that RELU is a partially linear function. Its first derivative is zero or one, and the second is always zero. The authors propose using sine as an activation function for implicit neural representations to address this problem.

$$\Phi(x) = \mathbf{W}_n(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \phi_0)(\mathbf{x}) + \mathbf{b}_n, \quad \mathbf{x}_i \to \phi_i(\mathbf{x}_i) = \sin(\mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i)$$

Where ϕ_i is an i-th layer of the MLP.

Since the sine function is not linear and the derivative of a sine function is cosine (phase-shifted sine function), it will have the ability to model the high-order derivatives better. It follows that the derivative of the sinusoidal representation network (SIREN or MLP with sine activation function) is itself a SIREN with all the network properties, and this does not hold for any other non-linearity. Exploiting SIRENs and a specific constraint F is possible to study not only the relationship between x and $\Phi(x)$ but also the relationship between x and derivatives of $\Phi(x)$, which can be beneficial in various problems.

With a specific weights initialization scheme that controls the distribution of activations, it is possible to build deeper networks and achieve faster convergence than with activation functions such as ReLU, RBF-ReLU, ReLU with positional encoding, and other non-linearities

One example of implicit neural representation is image fitting. Consider the case when needed to find a function that will map pixel coordinates on a grid to their RGB value.

$$\phi : R^2 \to R^3, \quad x \to \Phi(x)$$

$$D = \{x_i, f(x_i)\}$$

The dataset D used to fit the image consists of pairs of coordinates x_i and the pixel's value at that coordinate $f(x_i)$. The only constraint F that needs to be set is that function Φ must output a pixel intensity value based on $f(x_i)$ and itself, which can be translated into the loss function $\mathcal{L} = \Sigma_i ||\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)||^2$. The network, which was supervised using only pixel intensity values, represented the ground truth image and its derivatives. Considering other approaches, only the ReLU with positional encoding could represent ground truth image and in the representation of derivatives succeeded only SIREN.

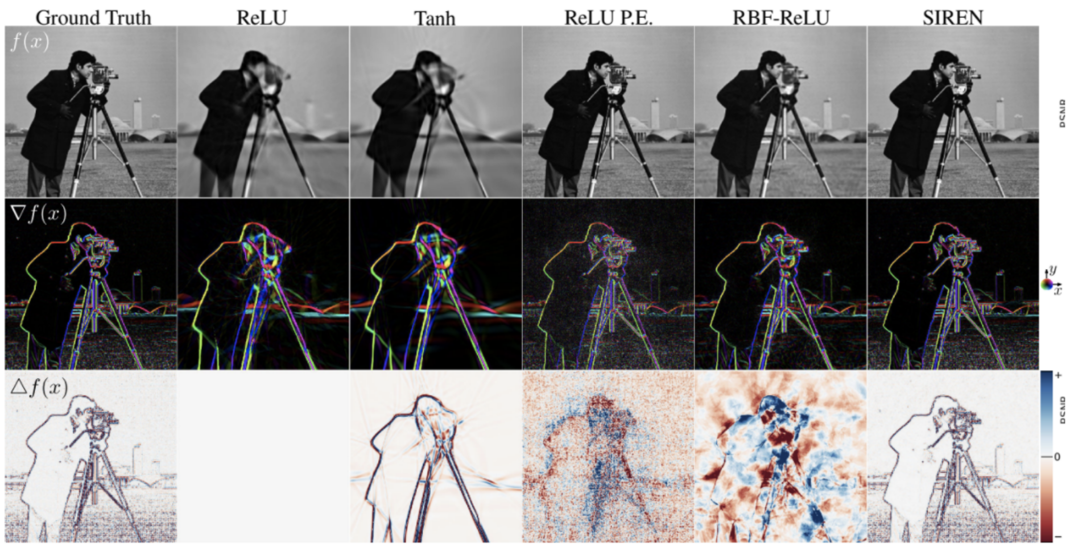


FIGURE 2.1: Comparison of different activation functions fiiting an image[41].

In this paper, the authors have shown many applications of implicit neural representations of signals such as image, video, and audio representation, solving first-order differential equations, 3D shape reconstruction, and learning space of implicit functions. The last application shows that the SIREN could not only learn one function but is also capable of parametrizing the whole space of functions. On the example of images, we can learn the global prior over images with a similar appearance and then use the parameterized space of functions for the image inpainting task. We decided to use this property of the SIREN in our work for anomaly detection on images.

2.2 Generative Adversarial Network

Supervised machine learning models are divided into two categories: discriminative and generative models. They approach the problem from two different angles. Discriminative algorithms directly model the conditional probability of the target Y given data samples X, $p(y|x)$. It finds the decision boundary between classes. Generative models are algorithms that make predictions by approximating joint probability distribution $p(x, y)$. Afterward, conditional probability $p(y|x)$ could be obtained by using Bayes Theorem. That is, they explicitly model the distribution of data. Discriminative learning algorithms are preferred in many tasks since they need fewer data and are more computationally efficient than generative algorithms. Probabilistic computations in Maximum Likelihood Estimation inside generative models are difficult to approximate.

Goodfellow et al.[13] proposed Generative Adversarial Networks (GANs), which tackle this problem by introducing a new way of training a generative model as a supervised learning problem that involves two models: a generative model G and a discriminative model D. GANs are unsupervised models that use supervised loss during the training process. Exploiting prior noise $p_z(z)$, most often sampled from Gaussian Distribution, generative model D learns distribution p_g over the data. Discriminative model $D(x; \theta_d)$ is responsible for classifying whether sample x came from data distribution or generative model distribution p_g[12]. $D(x)$ outputs the probability of whether x belongs to the p_{data}. Discriminative model is trained to maximize the probability of correctly distinguishing samples from p_{data} and p_g when Generator is trained to minimize $log(1 - D(G(z)))$.

In other words, generative model G and discriminative model D are playing a min-max game where they try to reach Nash equilibrium[33] with the following value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

The order in which we maximize the value function relative to the discriminator and then minimize relative to the generator is essential because we need to improve our generator relative to the best discriminator. During GAN training, it is necessary to have a balance between the discriminator and the generator; in terms of game theory, they need to be in equilibrium because if the discriminator trains faster and can distinguish samples from different distributions, then the generative model G will not be able to approximate data distribution p_{data} well.

2.2.1 Conditional Generative Adversarial Network

In Generative Adversarial Network without conditioning, we cannot control modes of data being generated. Mehdi and Mirza[32] proposed to condition any extra information y to direct the generation process to a specific mode depending on y. They called it the Conditional Generative Adversarial Network (CGAN). Adding conditioning yields the following GAN value function $V(G, D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y)))]$$

They trained CGAN on the MNIST [30] dataset where the property we want to control will be a digit from 0 to 9. With the help of CGAN, it is possible now to control the generation of samples with a particular mode during the test time.

In our work as additional information for the generation process, additional information was the features of the cropped image after the last layer of the feature extractor.

2.2.2 Improved training for GANs

Finding a balance between the discriminator and the generator during GAN training is not a trivial task. Salimans, Goodfellow et al.[36] highlighted the problems that arose during GANs training and suggested different approaches to tackle these problems. Training involves gradient descent to minimize the objective function, which does not guarantee the Nash equilibrium, and therefore training fails to converge. Simultaneous minimization of the cost function for the discriminator and the generator does not guarantee the algorithm's convergence. If the cost function of the discriminator is minimized, then the cost function of the generator can increase, and therefore the algorithm may not converge in many cases. Second-way GAN training that can fail is the situation where discriminator gradients point to similar directions. If the generator produces a very plausible output, then it can continue to generate only this output.

In this paper, the authors showed many approaches to stabilize the training and improve the convergence of GANs, one of which is feature matching which prevents the generator from overfitting on the operating discriminator. They introduce a new objective that pushes the generator to match the expected value of the features on intermediate layers of the discriminator. The new objective for the generator is defined as:

$$||\mathbb{E}_{x \sim p_{data}} \mathbf{f}(\mathbf{x}) - \mathbb{E}_{z \sim p_z(z)} \mathbf{f}(G(\mathbf{z}))||_2^2$$

The second approach that they introduce is minibatch discrimination, which tackles the GAN problem of mode collapse. Since the discriminator processes each example separately, there is no relation between the gradients of these examples, and they can point in analogous directions which cause mode collapse. To avoid the collapse of the generator minibatch discrimination is used. This is a technique where the discriminator looks at several examples in combination, not separately. Additional information in the minibatch allows generating perceptually appealing images posthaste[36]. They also proposed an Inception score for measuring the quality of the images and other minor improvements for the GAN training. We experimented with some of them in our work.

2.2.3 Wasserstein Generative Adversarial Network

To address the vanishing gradients problem and mode collapse, Martin Arjovsky et al.[4] proposed to exploit Wasserstein distance instead of Kullback-Leibler divergence[28]. Wasserstein distance is intractable in the original form, but using Kantorovich-Rubinstein duality, it can be simplified to:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_\theta}[f(\mathbf{x})]$$

Where $\mathbb{P}_r$, $\mathbb{P}_\theta$ are two distributions and f is the 1-Lipschitz function that satisfies the following constraint:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|\,.$$

In order to calculate Wasserstein distance f is learned by a discriminator network D without any activation function at the end and single scalar output. Function f acts as a critic and evaluates the generated image. The higher the rating, the closer the generated image is to the real one. Without any additional constraint on weights, discriminator D could output any real number. In order for the function to satisfy the 1-Lipschitz conditions, they clip the weights after each gradient update to the range [-0.01, 0.01]. In later work, gradient penalty was proposed to satisfy the 1-Lipschitz restrictions. We experimented with different GAN architectures during our work on anomaly detection on images, including Wasserstein GAN.

2.3 Anomaly Detection

Anomaly detection is a problem of identifying non-conforming items in the dataset. These non-conforming items are often called outliers or anomalies. Their appearance may indicate objective internal changes or external interventions. Outlier detection has various use cases across different industries, including monitoring data-centers monitoring infrastructure, medical signal analysis, transaction monitoring, and spam detection. In almost every quantitative discipline, applications for outlier detection could be found. There are three main classes of approaches to tackle this problem.

2.3.1 Supervised anomaly detection

One class of anomaly detection approaches is supervised learning. To do this, we should possess a labeled dataset of normal and abnormal observations. Outliers are infrequent events that make up a small fraction of the data relative to standard samples. For anomaly detection with supervised learning, binary classification is performed. Most classification machine learning algorithms are made under the assumption that we have an equal number of samples in each class. This results in bad performance of the classification algorithm, especially for the poorly represented class. Impaired performance in a minority class is a problem because the anomaly detection algorithm aims to classify unusual samples correctly and therefore is more delicate to errors in the minority class. Besides, the labeled dataset could be non-representable of all the possible outliers. During training, the classification algorithm may see an anomaly not at all similar to the training sample.

Also, abnormal samples on the test network may be from a different distribution altogether. A trained classification algorithm will not detect them, which will worsen the test results. Therefore most of the anomaly detection algorithms are either statistical or unsupervised.

2.3.2 Semi-supervised anomaly detection

Semi-supervised anomaly detection algorithms are using a dataset with only regular samples without any outliers. The algorithm learns the general data model and then determines the anomalies by the substantial deviation from the data model. These methods are also called one-class classifiers. One of the most famous algorithms for semi-supervised anomaly detection is One-Class SVM[39]. During training, SVM sees only data without anomalies, and then this model classifies anomalies and standard samples on test data. It tries to alienate data instances from the origin in kernel space, leading to complex structures that describe the dataset. One-Class SVM tries to find a function that is positive for areas with high density and negative for low density. Although One-class SVM is a semi-supervised method, by using soft-margin, it becomes unsupervised.

Approaches based on the autoencoders[27] suggest that it can learn similar patterns in a normal signal, and it will be able to reconstruct them well. Autoencoders are a class of neural networks which goal is to produce output most similar to the input. They compress input into the latent space and then reconstructing the output from this low-dimensional representation. Autoencoders suggest that if the model is learned to reconstruct normal data, it will not reconstruct the anomaly.

2.3.3 Unsupervised anomaly detection

The most flexible setup for anomaly detection, which does not require additional information, is unsupervised learning. These approaches do not differentiate between training and test samples. They detect anomalies based only on the internal properties of the data. Since anomalies are rare and unknown during training, the problem is reduced to modeling the distribution of regular data and introducing measurements in space to detect anomalies. Unsupervised anomaly detection has three major classes: Nearest-neighbor methods, Clustering methods, and Statistical algorithms.

The most straightforward way of anomaly detection is k-nearest-neighbors[3] which is different from the k-nearest-neighbors classification algorithm. This method is designed to detect global anomalies and is not capable of detecting local anomalies. Initially, the distance to the k nearest neighbors is calculated for each point, and then an anomaly score is calculated. Based on the variation of the algorithm, the distance to kth-NN or the average of k-NNs is taken. Value of the anomaly score is highly dependent on the distance we choose and the data normalization. Therefore is not a trivial task to select an appropriate threshold for outliers.

The second commonly used anomaly detection technique is clustering-based, which groups regular training data into clusters. If the new points do not belong to any of the clusters, they are considered anomalous. By type of clustering, algorithms are divided into single and multi-class methods. DBSCAN is most often used to detect anomalies with the help of clustering[10]. For each point, the neighbors in a particular neighborhood with a radius of ϵ are counted. If this value is greater than the minimum number of points, then a cluster is created, and this process continues on the neighbors' neighbors. Then for a new point, distance to core points of the cluster is calculated, and if this distance is more significant than ϵ, the point is considered anomalous. K-means[24] and other clustering methods, such as hierarchical clustering, are less preferable but could work in specific cases.

Third anomaly detection technique purposes estimating the probability density distribution of the normal data. Density estimation on a training dataset is a straightforward method of anomaly detection. A new point that lies in an area of high density is classified as normal, in another case, it is an anomaly. The most common methods used in this class of algorithms are Kernel Density Estimation and Gaussian Mixture Model. Gaussian distribution is often used to model the data. A more versatile approach is the Gaussian Mixture Model which is a linear combination of normal distributions with the following formula for probability density:

$$P_{GM}(x) = \frac{1}{N_{GM}} \sum_{j}^{N_{GM}} \left\{ a_j \frac{1}{(2\pi)^{d/2} \left|\Sigma_j\right|^{1/2}} \exp\left\{ -\frac{1}{2} \left(x - \mu_j\right)^T \Sigma_j^{-1} \left(x - \mu_j\right) \right\} \right\}$$

Means and covariances of the single Gaussian component are optimized using the Expectation-Maximization algorithm.

2.3.4 Anomaly detection in computer vision

Anomaly detection is a huge problem that occurs in various fields. GANs have successfully been applied to model high-dimensional data distribution. Generative adversarial networks can model normal data behavior using the adversarial learning process and then detect anomalies by measuring anomaly score [37]. The vast majority of GANs approaches to anomaly detection are based on an idea proposed by [8]. GAN maps samples from noise prior to data distribution in the original formulation, and the discriminator tries to distinguish between generated and actual samples. In this work, Bidirectional Generative Adversarial Network (BiGAN) was introduced. BiGAN extends this training process by adding the inverse mapping of the output data back to latent space. This strategy is a basis for anomaly detection using GANs. Schlegl et al. proposed AnoGAN[38], a deep convolutional generative adversarial network for anomaly scoring strategy using the mapping from image space to latent space. Efficient GAN-Based Anomaly Detection (EGBAD) was introduced as a first approach using BiGAN architecture[44]. EGBAD allows computing anomalies without optimization step like in AnoGAN. Inspired by BiGAN, AnoGAN, and EGBAD, Akcay et al. introduced the new approach GANomaly[2]. They trained Autoencoder to learn low dimensional representations simultaneously with GAN for learning the manifold of regular samples. This approach was an improvement because it requires nothing but a generator and discriminator network as in the original method.

Some of the recent studies proposed alternative anomaly detection approaches based on image completion. Haselmanm et al., in two consecutive works[18][17], proposed deep learning based approach for anomaly detection on images. In the first paper, a convolutional neural network (CNN) is used for image completion on the image patches to reconstruct its pattern. Since the network is trained only on normal image patches, it should reconstruct them well. Then by taking a pixel-wise difference of input and output anomaly score is obtained. In the second work, they used a fully convolutional network (FCN) for defect segmentation. They proposed to generate artificial defects on normal data and then train the model in a supervised setting.

Chapter 3

Problem Formulation

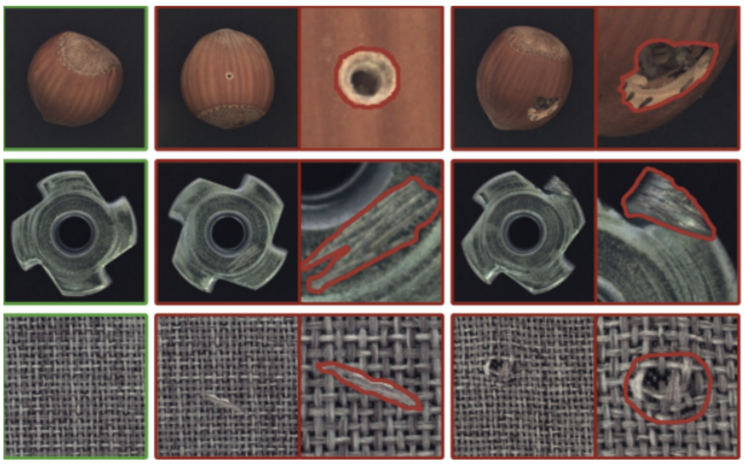


FIGURE 3.1: Example of image anomaly detection[5]

People are very good at recognizing whether they have seen this image before, whether it is something new or abnormal. So far, machine learning approaches fail to detect anomalies among ordinary objects well enough. There are many applications where detecting anomalies will be helpful. One of the critical factors for optimization in production is the automatic detection of anomalies. We can recognize outliers by visual inspection of an object and search for non-conforming patterns on it. For humans, this does not cause any problems, but it is pretty tricky for computers. Unlike other computer vision tasks, anomaly detection suffers from many problems such as class imbalance, different appearance of anomalies, and undiscovered anomalies. Anomalies are infrequent. In addition, they can be of different textures, sizes, and shapes. Different appearances make it very difficult to collect data in order to cover all types of anomalies. Some algorithms are designed in a semi-supervised setting. They are trained using only standard samples.

Recently, deep learning is widely used in the field of computer vision. Deep learning algorithms are capable of modeling complex non-linear data patterns and acquire unbelievable results in many areas. Deep generative models as one part of deep learning could model the distribution of the data. Generative Adversarial Network is an essential branch in deep generative modeling. This algorithm can learn to model data distribution and reproduce it. Since GANs can learn to represent data distribution, some recent studies exploited it for anomaly detection. GANs are the most potent generative models which generate the most plausible results. GANs combined with the latest advances in computer vision could bring anomaly detection techniques to a higher level.

The signal can be presented with the help of different representations. A new approach to representation was proposed by Sitzmann et al., in which a fully connected neural network with periodic activation function (SIREN) is used to parameterize the image. This approach can model the signal in great detail and its derivatives. They also showed that with the help of Hypernetwork, it is possible to generalize over the space of SIRENs, which makes it possible to solve inverse problems like image inpainting. If we can learn such a prior over SIRENS that parameterize a set of images, it can be used in semi-supervised anomaly detection, where we learn the appearance of normal data.

In the scope of this study, we rely on SIREN as a primary approach for image reconstruction. We also used many techniques to directly train SIREN via an adversarial training framework combined with loss functions for image quality improvement. Summarizing, in this work, we focus on the following directions:

- Impact of Triplet Loss as a loss function for discriminator network D.
- Investigate the Sinusoidal representation network for the anomaly detection task.

Chapter 4

Method

Our approach has the following essential components: Objective Function, Crop convolution decoder, Periodic activation for image reconstruction, and Discriminator as anomaly estimator, described in the corresponding sections below. We focus on different approaches to improve the reconstruction capabilities of our method. We study combinations of different loss functions and various architecture variations. Besides, we tested embeddings of the trained discriminator with Triplet loss for anomaly score calculation.

Four our method training, we experimented with a combination of several loss functions. Our primary loss for the reconstruction was the Mean Squared Error (MSE). With the help of MSE, the Generator can learn a good reconstruction on a training set but still could have bad perceptual quality on a validation set.

4.1 Network architecture

Our approach is a classical auto-encoder image-to-image translation network. From now on, we will call it a Generator (G). Where Decoder (Dec) is based on SIREN with different variations in architecture. One of the branches in this study is training Dec as a GAN with different setups of the Discriminator (D). The goal is to train G that takes the image with the cropped center(I^C) as an input to Encoder (Enc), Enc learns data representation in the cropped region based on patterns from the area around the crop and removes image noise, Dec reconstructs cropped part of I^C.

When we train G as a GAN, D takes the output I^R and compares it with the entire input image I^I. Training should be done on a set of images without anomalies. After training, G is able to reconstruct cropped area as it is normal, and D can evaluate a level of abnormality after compassion between normally reconstructed I^R and abnormal I^I. We also study the reconstruction capabilities of the Generator without an additional discriminator evaluation during training.

4.1.1 Crop convolution encoder

Generator G takes an image I^I with a cropped center and tries to reconstruct the missing region. To train the SIREN on a set of images, we need to provide supplementary information to the input. In our work, as additional information, we used high-level feature representations obtained from the convolutional neural network encoder (*Enc*). If we feed an image with a cropped center to *Enc*, then closer to the last layer, most of the values in the filter responses will be zero. To overcome this problem, we propose to use *Enc* with Crop Convolutions instead.

Crop convolution works the same as the regular convolution(conv.) layer. It slides through the height and width of the image with a given stride producing the image representation of the given receptive field. We changed the convolution in such a way that when the kernel slides through the image, it touches the cropped center only by a certain number of pixels. During our study, we experimented with activation functions for our encoder. We used the sine activation function with the initialization scheme described in Section 4.1.2. This experiment did not give improvements in both the reconstruction quality and training speed. Therefore we decided to use ReLU as an activation function for our *Enc*. As a weights initialization, we used Kiaming Initialization[19] which considers the non-linearity of the ReLU activation function.

During the training, the input to our Enc is a fixed size 64x64 grayscale image. The image is passed through a stack of crop conv layers, where filters with a receptive field of size 3x3 are used. The stride of the convolution is fixed to 1 pixel. The spatial padding of each crop convolution layer is 1. This is done in order to reduce the original size by two times after each layer. Crop convolution layers are followed by Fully-connected (FC) layers which have 512 channels. FC was added to diminish the size of the feature map, which reduced the input size to the SIREN several times. Many experiments have been conducted with the last layers of the encoder to give the decoder input the most expressive vectors. The best results were obtained with 1 FC layer with 512 activations.

To standardize the inputs to layers for each minibatch, we use batch normalization after each layer. It helps to reduce the number of epochs needed to train the Enc and stabilize the training process[23].

4.1.2 Sinusoidal representation network

With the SIREN, we can obtain a continuous representation of the signal (image), and it introduces a new method for solving the inverse problem like image inpainting. It is possible to fit a continuous SIREN representation by using only partial observations of the discrete signal. Many image processing problems could be solved by using this property.

As it is described in Section 2.1, to parametrize a single image with the SIREN, we need a dataset D. In order for SIREN to be able to parameterize the whole dataset of images, we added latent low-dimensional code to each pair of coordinates x_i in dataset D. We experimented with different sizes of latent code vector.

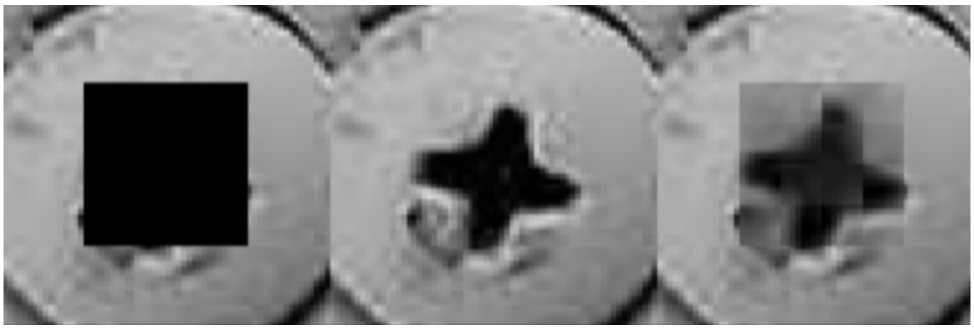


FIGURE 4.1: Example of JPEG-like artifacts

The function described in Section 4.2 can also act as a constraint for the SIREN. Style and perceptual loss also ensure that the output pixel intensity depends on the intensities of the original image $f(x_i)$. To parametrize an image, we need constraint C. It has been shown experimentally in Section 5.6 that a combination of perceptual, style and MSE losses can be also a good constraint for SIREN.

The final SIREN architecture during our study consists of 5 fully connected (FC) layers with 512 activations. To stabilize gradient flow and create deep architectures, SIREN needs to be initialized with control over the activation distribution. A unique weight initialization scheme was designed by Sitzmann et al., which ensures that at the input to each sine activation is normally distributed with a standard deviation of 1. Weights are drawn from the distribution:

$$w_i \sim \mathcal{U}(-\sqrt{6/n}, \sqrt{6/n})$$

First layer of the SIREN is initialized with weights so that the sine function spans multiple periods from [-1, 1]. The frequency of the first sinus activation was experimentally selected and is equal to 30. This initialization scheme grants SIREN fast and robust convergence using Adam optimizer[26].

$$\sin\left(\omega_0 \cdot \mathbf{W}\mathbf{x} + \mathbf{b}\right)$$

4.1.3 Discriminator as anomaly estimator

For the image abnormality evaluation, we exploit a discriminator trained with the Triplet loss function, which is often used in metric learning tasks[21]. Triplet Loss is a distance-based function with the following three inputs: anchor and positive are of the same class and negative, a different class from the anchor.

First, we calculate embeddings for every of three inputs by feeding them to discriminator D, whereafter the Triplet loss function is calculated. The goal of the loss function is to minimize the distance between the anchor (a) and positive (p) while maximizing it between the anchor and negative (n). Mathematically we can write triplet loss in the following form. Where d is a distance function used for calculation and margin m is manually selected.

$$\mathcal{L} = \max(d(a,p) - d(a,n) + m, 0)$$

Margin helps in the following two situations. In the first situation, the distance to the positive and negative is the same, and we need the distance to be less to the positive class. In the second case, the distance to the negative class is greater, but we still want to increase it slightly. Margin is used for such cases and is selected concerning the task.

During generator G training, the discriminator acts as an adversary. It distances the embeddings of the reconstructed image I^R from the ground truth image embeddings I^I and brings the I^I closer to the embeddings of the rotated image I^{Rot}. We had an assumption that in such a setting, the discriminator would be able to generate rotation invariant embeddings. It has been shown that adversarial training improves the result of image inpainting on some datasets. The loss function of the discriminator can be written as:

$$\mathcal{L} = \max(d(I^I, I^{Rot}) - d(I^I, I^R) + m, 0)$$

Image with an anomaly is used during test time as an input to the generator. We crop the center from the anomalous picture, and the generator must reconstruct it without an anomaly. Then cosine distance is used for the abnormality evaluation. By measuring the distance between the original image and reconstruction $\cos(I^I, I^R)$, we obtain an anomaly score. Final discriminator loss function can be written as :

$$\mathcal{L}_{\text{dis}} = \max(\cos(I^I, I^{Rot}) - \cos(I^I, I^R) + m, 0)$$

The discriminator is a three-layer CNN with 4-th FC. The network has 5x5 convolution in every Conv layer. PReLU[20] was used after each layer as an activation function. To reduce the dimensions of the feature maps, we added max-pooling after each layer. Also, it makes the network more robust to the changes in the feature positions in the image. To reduce overfitting and improve generalization, we used the Dropout layer[42] with p=0.2.

4.2 Objective

Reducing the MSE does not guarantee that the reconstructed region will have a flawless transition to neighboring pixels. Therefore the objective function of our method inspired by Guilin Liu et al.[31] is a combination of multiple loss functions. Given an input I^I and reconstructed I^R, we compute MSE, which in addition is also a perfect constraint for the SIREN-based encoder, which is described in Sec 4.1.2. For this purpose, any pixel-wise loss would be suitable, but we have empirically found that the MSE behaves best of all.

$$\mathcal{L}_{\text{mse}} = \sum_i \left\| I_i^I - I_i^R \right\|^2$$

To calculate the next pair of losses, we exploited the VGG-16 network proposed by Karen Simonyan and Andrew Zisserman as a loss network[40]. VGG-16 consists of 16 layers, including thirteen convolutions with 3x3 filters and fully connected layers. Stride and padding of all layers are fixed to 1 pixel. For further purposes, features only from convolution and pooling layers were used.

Convolution Neural Network (CNN) learns spatial hierarchies of features via backpropagation. If we feed an input image I^I to the CNN, it will be encoded in each layer by the filter responses. The perceptual loss function computes the difference of feature representations of I^I and I^R after each convolution layer[25]. Let Ψ_p be the activation of pth layer of the CNN. We can define perceptual loss as follows:

$$\mathcal{L}_{\text{perceptual}} = \sum_{p=0}^{p-1} \left\| \Psi_p^{\mathbf{I}^I} - \Psi_p^{\mathbf{I}^R} \right\|_1$$

The perceptual loss function is used to compare the style and context of two images. Perceptual loss is very similar to the per-pixel loss functions, and it is more often used in computer vision applications. In our case, it improves the quality of the reconstruction for some experiments described in Section 5.6. We use the VGG-16 network pre-trained for image classification on ImageNet[7] to calculate the perceptual loss.

As the last component for our objective function, we use style loss[25]. Style loss is similar to perceptual loss, but we construct a Gram matrix before calculating L1 distance. To determine the style of the image concerning the context, we need to calculate the gram matrix. We construct a gram matrix by computing the correlation of feature representations on different layers.

$$G_p^I = (\Psi_p^{\mathbf{I}})^\intercal (\Psi_p^{\mathbf{I}})$$

Where G_p^I is a product between vectorized feature maps on layer p. If we calculate the same gram matrix for ground truth and reconstruted image, two can be compared to calculate the difference in style between I^I and I^R. Style loss is defined as follows:

$$\mathcal{L}_{\text{style}} = \sum_{p=0}^{P-1} \frac{1}{C_p C_p} |K_p (G_p^{I^I} - G_p^{I^R})|_1$$

Suppose feature representations on the pth layer have the shape $H_p \times W_p \times C_p$, the gram matrix will be of shape $C_p \times C_p$. K_p is a normalization factor which equals $1/(H_p \times W_p \times C_P)$.

The final generator objective is a weighted sum of the previously described loss functions with experimentally selected coefficients. It has the following form:

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{mse}} + 0.01\mathcal{L}_{\text{perceptual}} + 0.1\ \mathcal{L}_{\text{style}}$$

Generator and discriminator are trained with a joint loss function combined with a previously described generator loss function and discriminator loss function from Section 4.1.3. The objective for full network training is combination of generator and discriminator objective functions.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{dis}} + 10\mathcal{L}_{\text{gen}}$$

4.3 Anomaly score

Since the final discriminator is trained with triplet loss with cosine distance, we can determine the anomaly score by measuring the distance between I^I and I^R discriminator embeddings. Further, we need to determine the threshold, which will determine whether there is an anomaly in the image. To obtain cosine distance we need to subtract the cosine similarity from one. For two vectors, cosine similarity is given as follows:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

$$\text{distance} = 1 - \text{similarity}$$

Chapter 5

Experiments

5.1 Implementations details

We used a standard toolkit for Machine Learning. Solutions and experiments described in this paper were implemented on Python 3.8 (Van Rossum and Drake, 2009)[43], which interpreted high-level, general-purpose programming language. For the construction and training of the neural networks, PyTorch 1.5 (Paszke et al., 2019)[34] was chosen over TensorFlow (Abadi et al., 2015)[1], as it is more intuitive and works on dynamic graph concept. The whole pipeline is constructed using PyTorch Lightning[11] since, once implemented, it is possible to add new functionality rapidly. For the visualization of the learning process, Tensorboard was used because it is effortless to plot losses and images. In order to process images, manipulate with matrices and do visualizations, we employ Opencv[6], NumPy[16], and Matplotlib[22].

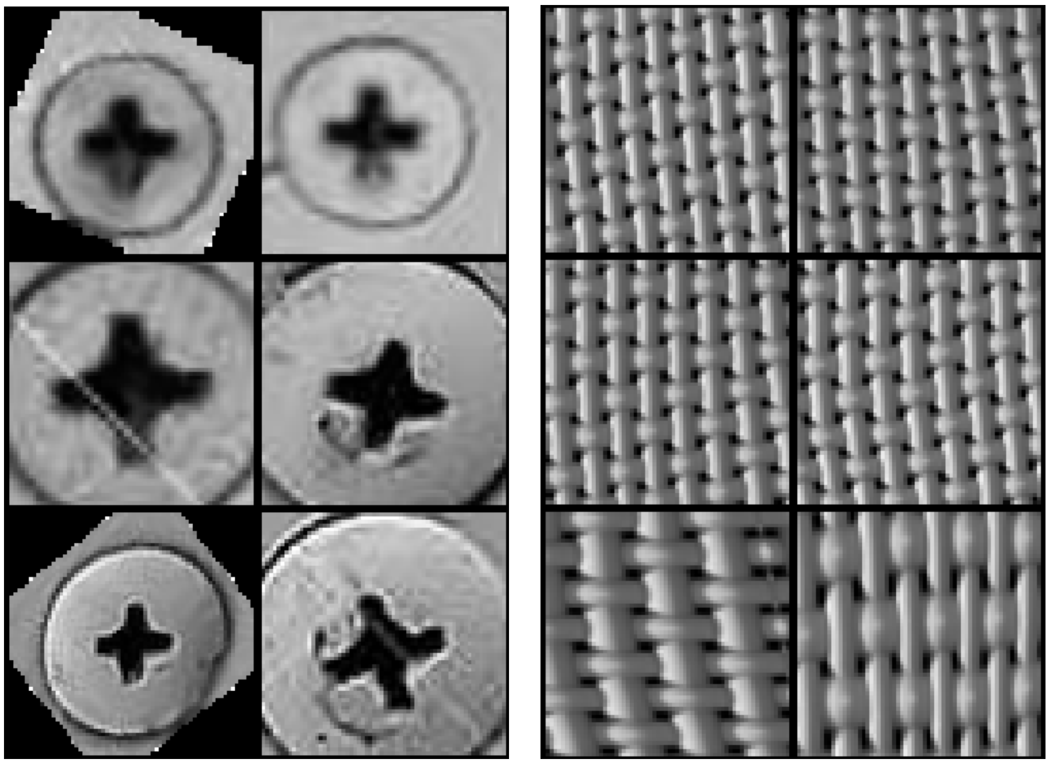
This section will provide an overview of the experiments that have been performed. Calculation of the abnormality score will be conducted only for the best experiments on two datasets.

5.2 Datasets

Two primary datasets were used to train the generator. The first dataset consists of aircraft rivets. Rivets are used to connect two metal sheets on an airplane. Counting and inspecting the rivets is a crucial step in inspecting the aircraft. The first task we faced during our study was to find out if there was an anomaly in the image of the rivet. The dataset used in our work was assembled independently and consisted of 80 rivets of the same type. They were collected from the images of the aircraft's skin. Many experiments have been conducted to select the correct sample to train our algorithm. We decided to return all the rivets so that the orientation of the cross coincided. An example of rivets is illustrated on Fig 5.1

In order to test the ability of the generator to capture patterns, we tested it on the Pattern Dataset[29], which consists of big-scale images of 32 different knit patterns. The network was trained on 64x64 patches cropped from a single image of this dataset. We applied Elastic Transform to the image before cutting the patches so that the patches did not look the same during training and were curved (Fig 5.1). This dataset has been used in recent experiments with and without a discriminator.

Both datasets were divided into three parts train 60%, validation 20%, and test 20%. Artificially generated anomalies were added to the test sample to calculate the anomaly score.



(A) Aircraft rivets dataset (B) Texutres dataset

FIGURE 5.1: Sampels from two main datasets used during our study.

During the discriminator training, as we described in Section 4.1.3, we bring the embedding of the original image closer to the rotated one. The rotated images were obtained through a random rotation taken from the interval [-90, 90] degrees. The generator was trained on similar images because this method can only fit images with similar global context (i.e., patterns or objects).

5.3 Training details

All generator variations were trained on NVIDIA RTX 2080, using Adam optimizer for all network parts. We use different learning rates for the discriminator and other generator parts equal to 0.0001 and 0.000005, respectively. Since Adam[26] is recommended to use for SIREN training and we found no meaningful reason to use another algorithm. All the parts of the generator except the SIREN encoder are initialized using Kaiming Initialization[19]. SIREN initialization algorithm is described in Section 4.1.2

5.4 Experiments with Generator

Initially, experiments were performed with convolutional layers in the encoder. Since the center of the image was cut out, there were many zeros in the final feature maps during the convolution process. It worsened the final reconstruction of the generator. It was decided to use a crop conv layer in the encoder that affects the cut center only by a certain number of pixels. Thus, we obtained feature vectors that improved the reconstruction of the SIREN-based generator.

Many experiments with the encoder architecture have also been performed. First, we fed the generator vector from the last convolutional layer with a dimension of 4096 values. During training with this configuration, the generator took up more than 10 gigabytes of GPU memory. To reduce the size of the input vector, we added an FC layer with 512 channels. This reduced the overall size of the models by 4 gigabytes. By adding batch normalization, we accelerated the generator training. The final model was trained for 5 thousand epochs. Training it for a longer time leads to divergence and poor results on a validation set. Such a large number of epochs can be explained by the fact that the SIREN needs many iterations to fit a set of images. The best-performing model has good reconstruction results but lacks variability in the generated images and sometimes has a poor transition to neighboring pixels, as is illustrated in Fig 5.2.

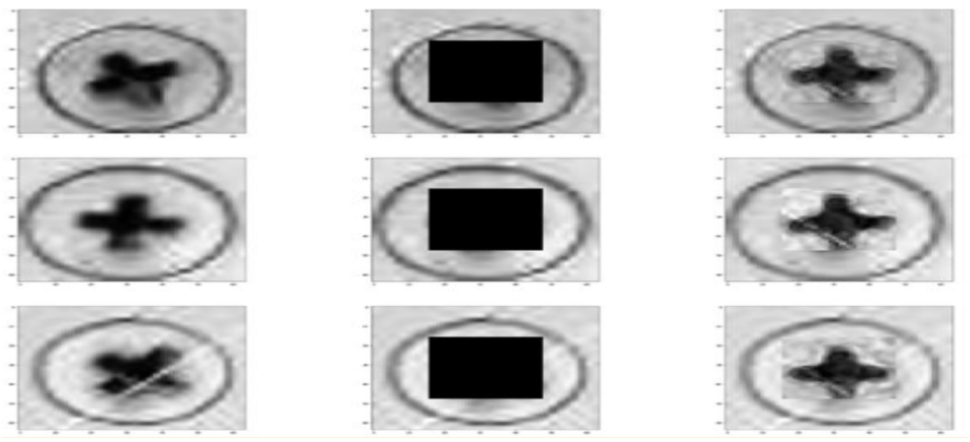


FIGURE 5.2: Example of generator reconstructions with the best parameters. From left to right: Ground truth, Cropped, Reconstruction

Experiments were performed with different sizes of the central cutout and data sets. We hypothesized that by turning all the rivets in one orientation, we would improve the reconstruction. We used damaged test images to look at the quality of the reconstruction. This experiment showed that the model is not able to generate images with good perceptual quality. We also used the loss function in Hasselman paper to train our generator. The generated images in both experiments were monotonous and of poor quality. These experiments are illustrated in Fig 5.3 and Fig 5.4.

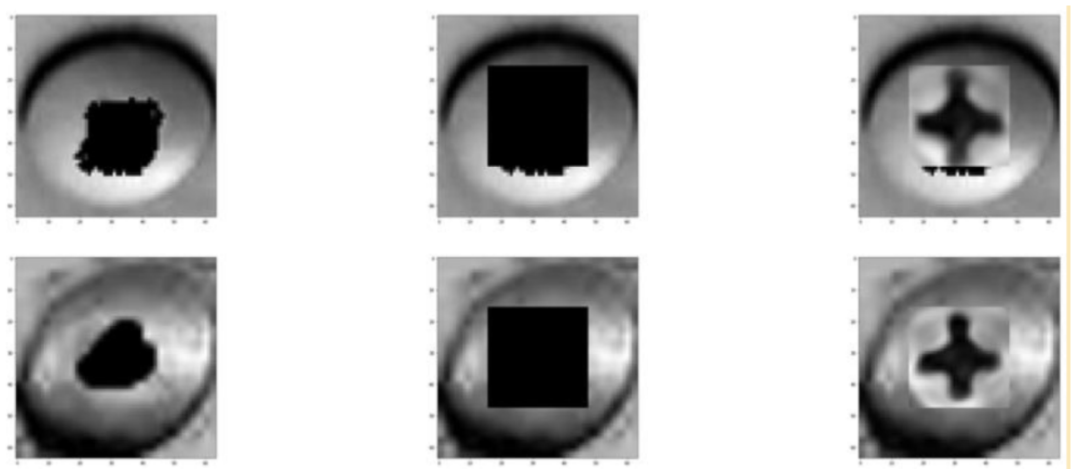


FIGURE 5.3: Reconstructions of the generator trained with convolutional layers in encoder.

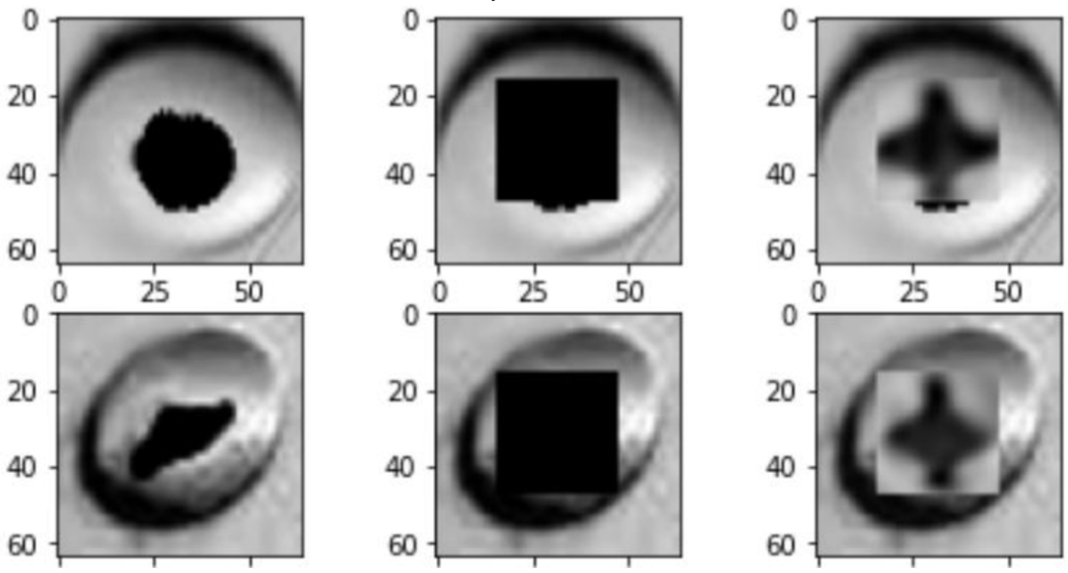
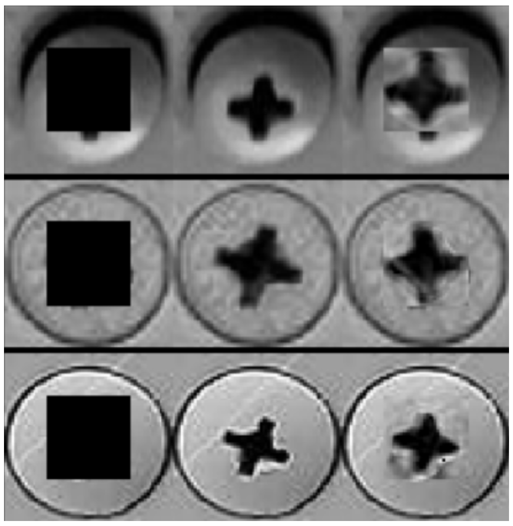


FIGURE 5.4: Reconstructions of the generator trained with Hasselman loss.

5.5 Discriminator

We added a discriminator to the architecture to increase the variety of images generated and make them less blurry. First, we utilize a deep convolutional discriminator first described in the DCGAN paper by Alec Radford Luke Metz[35] as a binary classifier that takes an image and outputs the probability that the input image is real or generated. The discriminator loss function, which is used in this experiment, is described in Section 4.1.3. Since the SIREN needs many iterations to learn, the discriminator learned faster and thus did not allow the generator to fit the images. We tried the different approaches described in Section 2.2.2 to balance the training of our network. The best reconstruction we could get is shown in Fig 5.5. To improve image quality and training stability (eliminate mode collapse), we utilize the WGAN discriminator delineated in Section 2.2.3. We used WGAN with a gradient penalty for weight clipping. This is an extension of the original WGAN. In such a setting, the discriminator acts as a critic and outputs a single scalar value. It is worth pointing out that models with discriminator were capable of fitting a train set of images perfectly but could not generalize well enough to perform anomaly detection. Reconstructions of the training set are illustrated in Fig 5.6.


(A) Experiment with WGAN discriminator

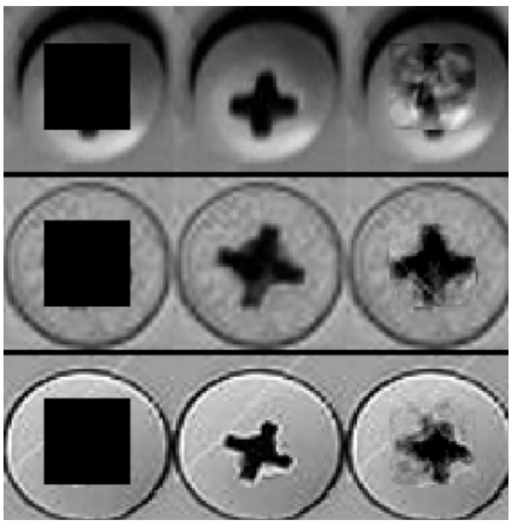

(B) Experiment with DCGAN discriminator

FIGURE 5.5: Experiments results of different discriminator architectures

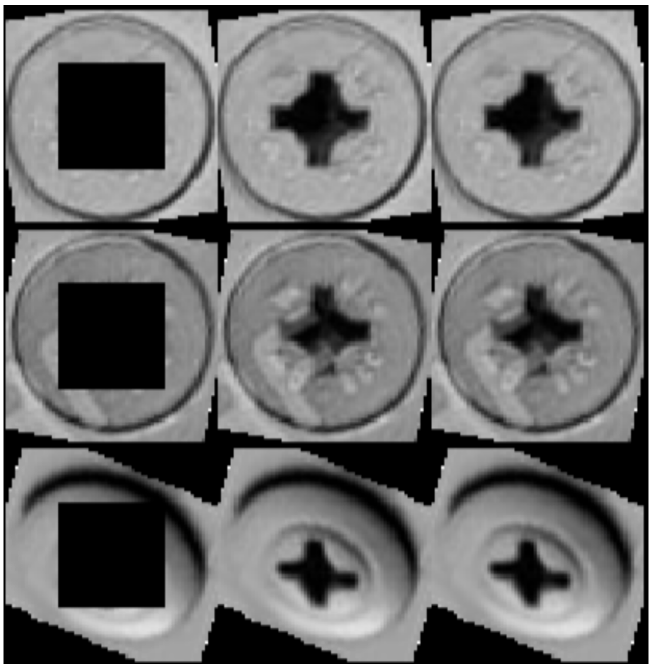


FIGURE 5.6: Example of reconstruction on the training dataset. From left to right: Cropped , Ground truth, Reconstruction

We trained a discriminator with the Triplet loss as an objective function. The goal was to calculate the abnormality score of the image during test time with discriminator embeddings. Two main distance functions used during our study are Euclidean and Cosine. During the experiments, we constructed different variations of Triplet loss:

1. $\cos(I^R, I^{Rot}) - \cos(I^I, I^{Rot}) + m$
2. Discriminator objective described in 4.1.3
3. Triplet with Euclidean distance

We changed both the dimension of the feature vector from the discriminator and the distance function used in the Triplet loss calculation to compare different configurations. We experimented with cosine distance in Triplet loss to obtain similar discriminator embeddings for original and rotated images. To compare each of these loss functions, we trained a model with experimentally best hyperparameters. Since the network had better reconstruction with cosine distance, we used it in further experiments.

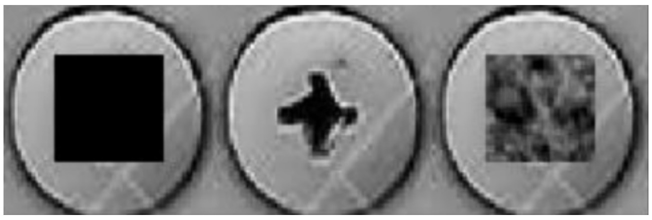
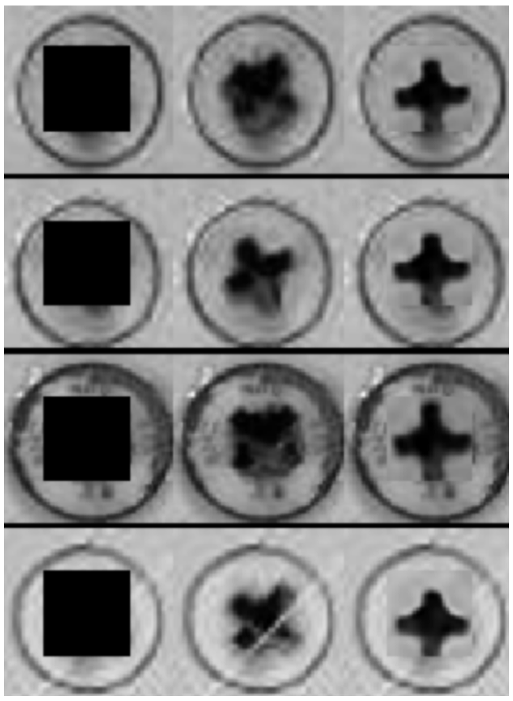


FIGURE 5.7: Results with first loss function


(A) Results with second loss function

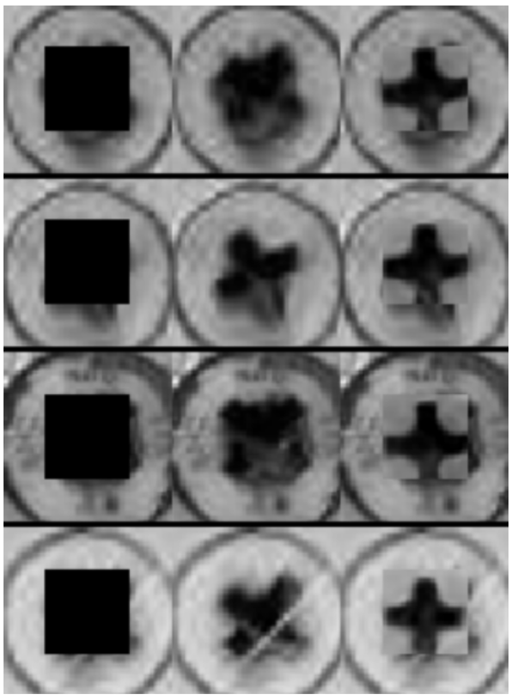

(B) Results with third loss function

FIGURE 5.8: Experiments results with different loss functions

In many cases, the discriminator trained much faster than the generator and did not allow the latter to create good reconstructions. We tried to train the generator once in a certain number of epochs and reduced the learning rate every few epochs with the help of Step Scheduler. In this setup, our model began to overfit after 2,000 iterations. We fixed the learning rate an order of magnitude smaller than the generator and took away the scheduler, which helped balance the training. The final learning rate for the discriminator is 0.0001.

We wanted to test whether the model is capable of recovering repetitive patterns. To do this, we used multiple images of knit patterns from the dataset described in Section 5.2. We fed a few repetitive patterns to the generator, and it tried to reconstruct the missing region. The result of the model can be seen in Fig 5.9.

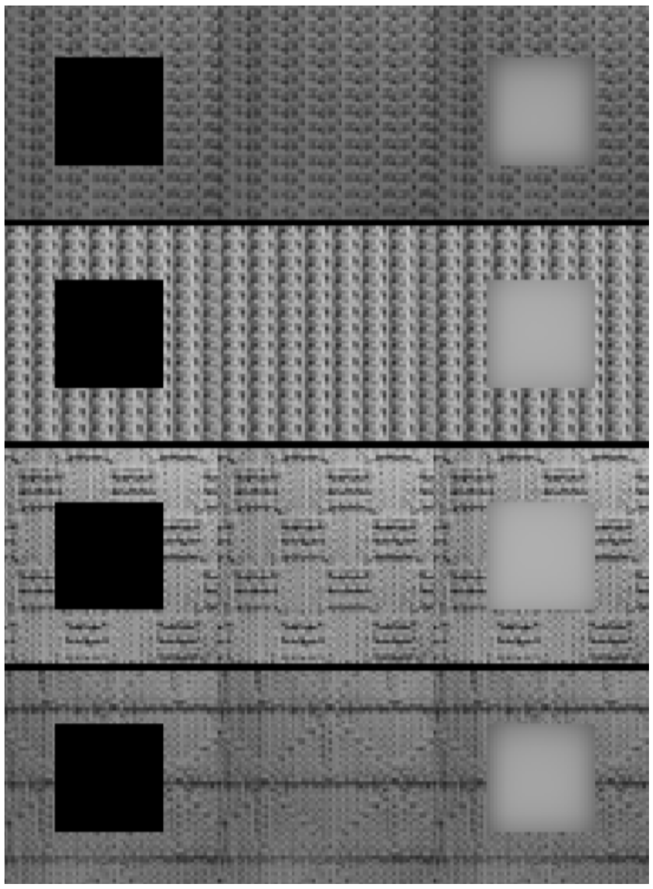


FIGURE 5.9: First attempt with textures dataset

5.6 Experiments with the objective function

The perceptual loss compares feature representations of two images on different representations levels of the Loss network. We wanted our generator to get information about the pattern present in the image and reconstruct it. However, additional information from perceptual loss was not enough to obtain reconstruction with the correct pattern and good perceptual quality. In order to catch the style relative to the context of the image, we added style loss. Experimentally selecting the coefficients, we obtained a reasonable reconstruction of the train set. The reconstruction on the validation data was very noisy and did not have a clear structure. Summing up these experiments, we can say that our model cannot learn a data set that is very different from image to image.

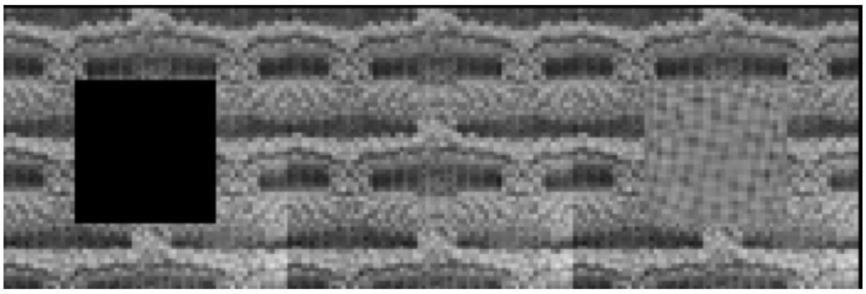


FIGURE 5.10: Trained with perceptual

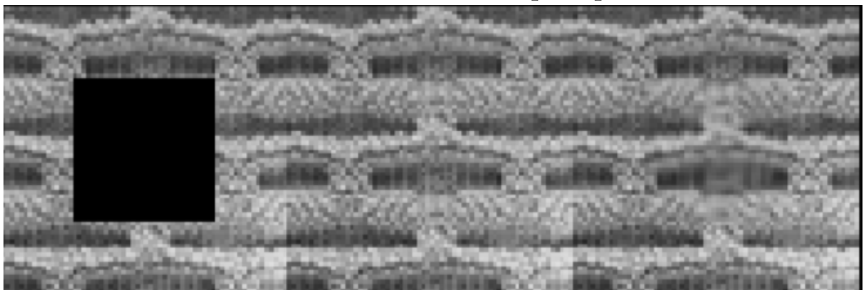


FIGURE 5.11: Trained with perceptual and style combined

FIGURE 5.12: Comparison of reconstruction only with different loss functions

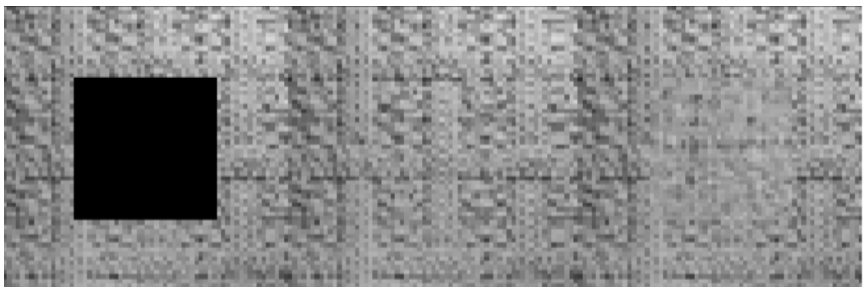


FIGURE 5.13: Single example from texture validation set

In order to train the model on similar patterns, we performed experiments on the data described in Section 5.2. Experiments have shown that the model can learn patterns in the image, but there is still room for improvement.

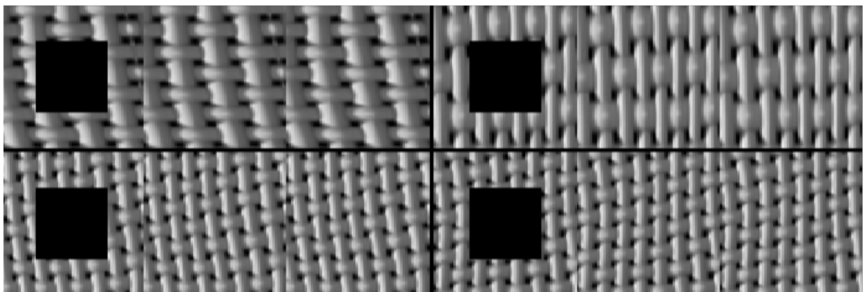


FIGURE 5.14: Reconstructions on the training set

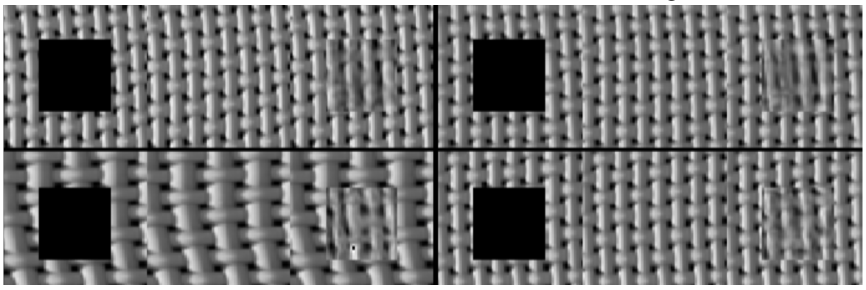


FIGURE 5.15: Reconstructions on the validation set

5.6.1 Loss Network

To calculate perceptual and style losses that measure perceptual and semantic differences between images, we make use of a pre-trained 16 layer VGG network[40] on ImageNet[7]. ImageNet consists of more than 14 million images with approximately 21 thousand classes of objects. Features extracted from the convolutional parts of the network were used to calculate the losses.

Chapter 6

Results

All the anomaly scores in this chapter are calculated with cosine distance between discriminator embeddings. To evaluate different experiments on a particular dataset, Discriminator weights were taken from the experiment with the best perceptual quality and variety of generated images. In this section, we will evaluate our generator's reconstruction based on the anomaly score on damaged test images of two distinct datasets. First, we will evaluate the reconstruction of rivets with and without a discriminator and then review the results on the texture dataset. The final models for all experiments were trained with the loss function from the section.

6.1 Results on the rivets dataset

As shown in Table 6.1 and Table 6.2, the reconstruction without a discriminator have worse results than with it. The combined training of the generator with the discriminator improved both the visual appearance of the reconstruction and the anomaly scores. The low anomaly scores can be explained by the fact that the center of the rivet itself is black, and therefore the distance between I^I and I^R is not so large. In such cases, we can use a small value for the threshold. It is also noticeable that the reconstruction with the discriminator is not so blurred and has a better transition to neighboring pixels.

6.2 Results on the textures dataset

From Table 6.3 we can see that the model can not consistently well restore the pattern present in the picture. The model lacks the information that comes from the encoder in order to recover the pattern correctly. The problem may also be with the small size of the input to the generator. In the general anomaly scores on the texture dataset are pretty high. Nevertheless, this is understandable because the model gives something in between several inputs.

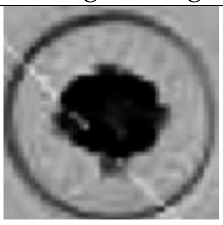
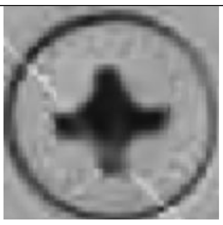
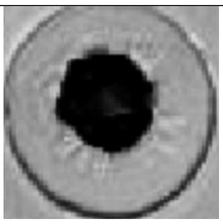
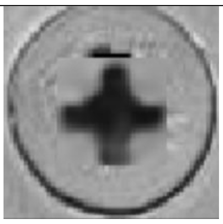
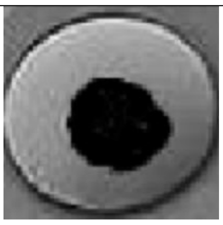
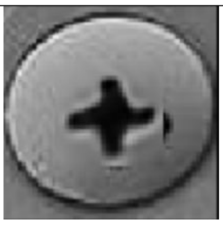
Damaged image	Reconstruction	Score
		0.427
		0.396
		0.598

TABLE 6.1: Anomaly score with reconstruction from generator

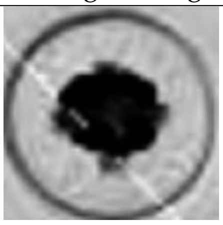
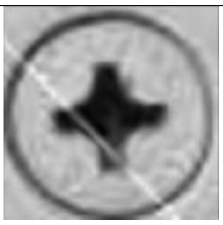
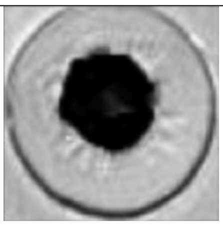
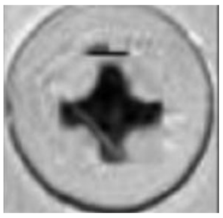
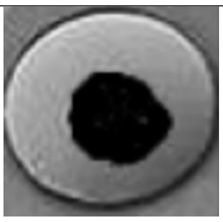
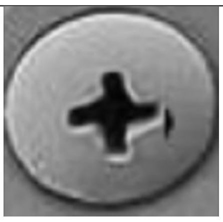
Damaged image	Reconstruction	Score
		0.415
		0.465
		0.623

TABLE 6.2: Anomaly score with reconstruction from generator with discriminator

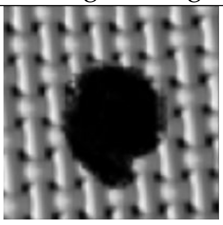
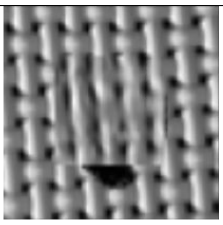
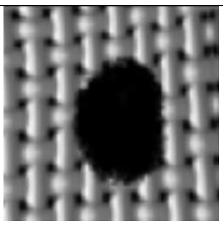
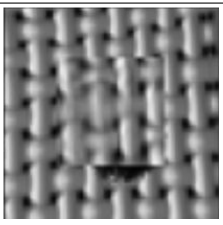
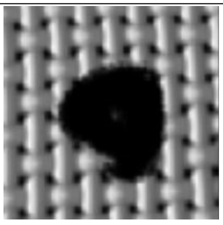
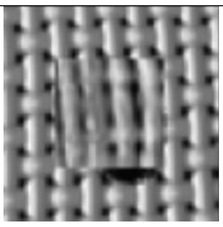
Damaged image	Reconstruction	Score
		0.978
		0.945
		0.939

TABLE 6.3: Anomaly scores on textures dataset

Chapter 7

Conclusion and Future work

7.1 Conclusion

In this work, we provide a method for unsupervised anomaly detection by using a sinusoidal representation network. We also investigated network training with a discriminator that uses a triplet loss function and showed its effect on the reconstruction of the missing image region.

We focused on the encoder part of the network and built our custom Crop convolution layer, which showed better reconstruction results than the standard 2D Convolution layer.

We examined discriminator architectures of different GANs, namely WGAN[4], DCGAN[32], and WGAN-GP[14], and found that discriminator with Triplet loss shows better results in image inpainting task.

We studied various losses and showed that a combination of Mean squared error, perceptual and style loss produce reconstructions with better perceptual quality. Also, training with a combination of loss functions improves texture reconstruction.

We showed that the sinusoidal representations network (SIREN) could be used in anomaly detection. Besides we propose an end-to-end training pipeline for the SIREN without the use of HyperNetworks[15].

In our opinion, the approach to anomaly detection with a sinusoidal representation network has great potential and must be explored further.

7.2 Future work

For the future work we consider the following directions. We want to explore how this approach behaves on different datasets and try our method for reconstruction of larger images. We also want to explore an approach where the missing region is not the center but a random part of the image. We plan to investigate whether our approach can be applied to a dataset with a large number of images with different context.

Improve feature representations that act as additional information for the decoder. Features from the Vision Transformer encoder[9] preserve global context in contrast to the features of the CNN. This will help in the reconstruction of the image with repetitive patterns.

Finally, we want to provide a broad comparison of existing unsupervised anomaly detection methods with our.

Appendix A

GitHub repository link: https://github.com/YuraYelisieiev/Anomaly-detection-with-sinusoidalrepresentation-network

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Samet Akcay, Amir Atapour Abarghouei, and Toby P. Breckon. "GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training". In: *CoRR* abs/1805.06725 (2018). arXiv: 1805.06725. URL: http://arxiv.org/abs/1805.06725.

[3] N. S. Altman. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". In: *The American Statistician* 46.3 (1992), pp. 175–185. ISSN: 00031305. URL: http://www.jstor.org/stable/2685209.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[5] Paul Bergmann et al. "MVTec AD - A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 9592–9600. DOI: 10.1109/CVPR.2019.00982. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Bergmann_MVTec_AD_--_A_Comprehensive_Real-World_Dataset_for_Unsupervised_Anomaly_CVPR_2019_paper.html.

[6] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[7] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[8] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial Feature Learning". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=BJtNZAFgg.

[9] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[10] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, 226–231.

[11] et al. Falcon WA. "PyTorch Lightning". In: *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* 3 (2019).

[12] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017. arXiv: 1701.00160 [cs.LG].

[13] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[14] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG].

[15] David Ha, Andrew Dai, and Quoc V. Le. *HyperNetworks*. 2016. arXiv: 1609.09106 [cs.LG].

[16] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[17] M. Haselmann and D. P. Gruber. "Pixel-Wise Defect Detection by CNNs without Manually Labeled Training Data". In: *Applied Artificial Intelligence* 33.6 (2019), pp. 548–566. DOI: 10.1080/08839514.2019.1583862. eprint: https://doi.org/10.1080/08839514.2019.1583862. URL: https://doi.org/10.1080/08839514.2019.1583862.

[18] Matthias Haselmann, Dieter P. Gruber, and Paul Tabatabai. "Anomaly Detection using Deep Learning based Image Completion". In: *CoRR* abs/1811.06861 (2018). arXiv: 1811.06861. URL: http://arxiv.org/abs/1811.06861.

[19] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123. URL: https://doi.org/10.1109/ICCV.2015.123.

[20] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *CoRR* abs/1502.01852 (2015). arXiv: 1502.01852. URL: http://arxiv.org/abs/1502.01852.

[21] Elad Hoffer and Nir Ailon. "Deep Metric Learning Using Triplet Network". In: *Similarity-Based Pattern Recognition - Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015, Proceedings*. Ed. by Aasa Feragen, Marcello Pelillo, and Marco Loog. Vol. 9370. Lecture Notes in Computer Science. Springer, 2015, pp. 84–92. DOI: 10.1007/978-3-319-24261-3_7. URL: https://doi.org/10.1007/978-3-319-24261-3_7.

[22] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[23] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[24] Xin Jin and Jiawei Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.

[25] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. "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.

[26] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[27] Mark A. Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE Journal* 37.2 (1991), pp. 233–243. DOI: https://doi.org/10.1002/aic.690370209. eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209. URL: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209.

[28] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79 –86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[29] Jonathan Leaf et al. "Interactive Design of Yarn-Level Cloth Patterns". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2018)* 37.6 (Nov. 2018). DOI: https://doi.org/10.1145/3272127.3275105.

[30] Y. LECUN. "THE MNIST DATABASE of handwritten digits". In: *http://yann.lecun.com/exdb/mnist/* (). URL: https://ci.nii.ac.jp/naid/10027939599/en/.

[31] Guilin Liu et al. "Image Inpainting for Irregular Holes Using Partial Convolutions". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*. Ed. by Vittorio Ferrari et al. Vol. 11215. Lecture Notes in Computer Science. Springer, 2018, pp. 89–105. DOI: 10.1007/978-3-030-01252-6_6. URL: https://doi.org/10.1007/978-3-030-01252-6_6.

[32] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

[33] John Nash. "Non-Cooperative Games". In: *Annals of Mathematics* 54.2 (1951), pp. 286–295. ISSN: 0003486X. URL: http://www.jstor.org/stable/1969529.

[34] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[35] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.06434.

[36] Tim Salimans et al. "Improved Techniques for Training GANs". In: *CoRR* abs/1606.03498 (2016). arXiv: 1606.03498. URL: http://arxiv.org/abs/1606.03498.

[37] Thomas Schlegl et al. *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. 2017. arXiv: 1703.05921 [cs.CV].

[38] Thomas Schlegl et al. "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery". In: *CoRR* abs/1703.05921 (2017). arXiv: 1703.05921. URL: http://arxiv.org/abs/1703.05921.

[39] Bernhard Schölkopf et al. "Support Vector Method for Novelty Detection". In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. Ed. by Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller. The MIT Press, 1999, pp. 582–588. URL: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection.

[40] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: `1409.1556 [cs.CV]`.

[41] Vincent Sitzmann et al. "Implicit Neural Representations with Periodic Activation Functions". In: *CoRR* abs/2006.09661 (2020). arXiv: `2006.09661`. URL: `https://arxiv.org/abs/2006.09661`.

[42] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.

[43] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[44] Houssam Zenati et al. "Efficient GAN-Based Anomaly Detection". In: *CoRR* abs/1802.06222 (2018). arXiv: `1802.06222`. URL: `http://arxiv.org/abs/1802.06222`.