technische universität
dortmund

Bachelor Thesis

Evaluating Anomaly Detection Algorithms by Generating Samples from Anomaly Scores

Ben Ernst
April 2025

Supervisors:
Prof. Dr. Emmanuel Müller
Simon Klüttermann

Technische Universität Dortmund
Department of Computer Science
Chair of Data Science and Data Engineering
http://ls9-www.cs.tu-dortmund.de

Contents

Acknowledgement

The authors gratefully acknowledge the computing time provided on the Linux HPC cluster at Technical University Dortmund (LiDO3), partially funded in the course of the Large-Scale Equipment Initiative by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as project 271512359.

Chapter 1

Introduction

Anomaly detection is a critical task in machine learning, with applications in many different fields, ranging from fraud detection to medical diagnoses [3]. Anomalies are data points that deviate significantly from the normal data, often indicating critical errors or opportunities to gain valuable insights. However, detecting anomalies is challenging due to their rarity and the complexity of the often high-dimensional data, leading to an increasing demand for anomaly detection methods that accurately identify anomalies. One way researchers have tried to improve those methods is by utilizing generative models to aid in the process of identifying anomalies [27, 33, 41]. While the use of generative models for anomaly detection has been extensively studied, there is limited research on how anomaly detection methods manage to generate data themselves. Studying their performance provides a novel way to evaluate and understand unsupervised models even in the absence of labeled data. This could also lead to insight into how well anomaly detection methods can learn the underlying normal data distribution, which is critical for improving their performance in real-world applications.

1.1 Structure of this Thesis

In the beginning, Chapter 2 provides a broader introduction into Anomaly Detection and some statistical methods of evaluating anomaly detection methods. Chapter 3 provides an overview of Generative Models, their challenges and applications. In Chapter 4 we will take a look at how Anomaly Detection and Generative Models relate to each other by highlighting related work as well as contribution. Chapter 5 describes the methodology used in this thesis, including the datasets, anomaly detection algorithms, the image generating techniques and our evaluation method. Chapter 6 presents the experimental results and analysis. Finally, Chapter 7 concludes this thesis and suggest directions for future work.

Chapter 2

Anomaly Detection

Anomalies are data points that are significantly different than the rest of the data [27]. They are not within the expected behaviour of the normal data [7]. Anomalies are rare and often indicate critical errors or opportunities. Those data points have many different names in literature, but the most common ones are *anomaly* or *outlier* [7]. Detecting those anomalies is a very important aspect in the real world, such as in detecting a hacked computer by detecting anomalous network traffic, detecting credit card fraud or other types of (financial) fraud, indication of malignant tumours in anomalous MRI images or in law enforcement [3, 7]. Rather than single data points anomalies can be comprised of multiple points at once. This set is called a *collective anomaly*. Single data points in a collective anomaly might not be an anomaly themselves. An example of that could be in fraud detection: a single large credit card transaction can be viewed as quite normal, but multiple large transactions in a short time frame will probably be classified as an anomaly. Another type of anomaly are contextual anomalies. Here a data point is considered an anomaly only in a specific context, but not in general. An example of such an anomaly could be in weather data: a very low temperature for instance would be classified as an anomaly when in summer, but not in winter. Applying an anomaly detection algorithm for contextual anomalies heavily relies on the meaningfulness of different contexts in the target domain as well as the availability of the attributes that define a context [7].

Fig. 2.1 shows a simple example of anomalies in a 2-dimensional data set. There are two normal regions, N_1 and N_2, since most data points are within those regions. Points that are far enough away from those regions could be considered anomalies, namely o_1, o_2 and O_3, which is a collective anomaly.

Anomaly detection algorithms can have two different types of outputs [3, 7]:

- **Anomaly scores**: In most cases an anomaly detection algorithm will return a score to each data point quantifying how much of an outlier the specific data point is. These scores can be used rank data points in order of their tendency to be an anomaly. This output retains all the information of the algorithm but does not give the user a number of points that should be considered anomalies.

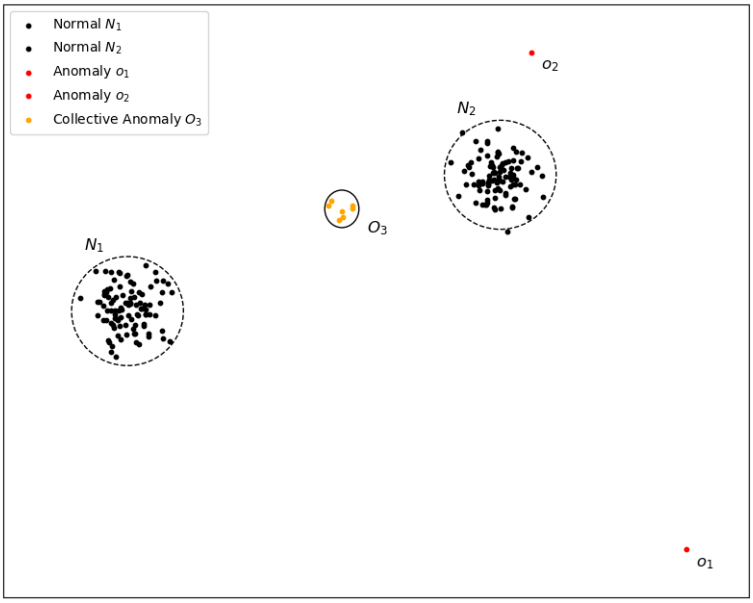

Figure 2.1: Basic example of anomalies

- **Binary labels**: Each data point is assigned a label (*normal* or *anomaly*). Some anomaly detection algorithms directly output these labels but they can also be created by applying a threshold to anomaly scores. This output contains less information than anomaly scores but it is often needed for decision making in practical applications.

Anomaly scores can be difficult to interpret. They are often just numerical values that provide little to no context about why a data point is considered more or less anomalous that another one. That is especially true for high dimensional data since there are many more features that could lead to the anomaly score being higher. Converting the anomaly scores into labels requires a threshold that can be subjective and context-dependent without a lack of guidance, amplifying the lack of interpretability. There are different ways to increase the interpretability of anomaly detection algorithm and it is an active part of research [39, 35].

Detecting anomalies can be done through various different approaches, including machine learning, deep learning or basic statistical methods. Deep learning is especially suited for anomaly detection in many applications since it can handle high-dimensional data and can learn complex patterns in the data. In this thesis we will focus on three deep learning-based methods, more specifically on Autoencoders, Deep SVDD and DEAN. We will take a closer look at those in Chapter 5.

Evaluating anomaly detection algorithms is usually done by using statistical methods. The most simple ones are *precision* and *recall*.

$$Precision = \frac{\text{Number of correctly identified anomalies}}{\text{Number of correctly and falsely identified anomalies}} \tag{2.1}$$

Precision is the proportion of points identified as an anomaly that actually are anomalies. Higher precision means fewer false flags but could lead to not as many anomalies being caught. A high recall on the other hand ensures that most anomalies are captured, while not necessarily minimizing false flags.

$$Recall = \frac{\text{Number of correctly identified anomalies}}{\text{Total number of actual anomalies}} \tag{2.2}$$

Recall is the proportion of actual anomalies in the set of all the points the algorithm labeled as anomalies. One way to ensure both minimizing false flags and catching most anomalies is to use the F1-score. It balances precision and recall using their harmonic mean.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.3}$$

The F1 score is especially useful when there is an imbalance between the normal class and the anomalous class because it considered both false positives (when the algorithm classifies data as an anomaly but it is actually normal) and false negatives (when the algorithm classifies data as normal but it is actually an anomaly) simultaneously. Since anomalies are by definition rare, using the F1-score in anomaly detection tasks can be a good choice. Another metric commonly used is the ROC AUC score [3, 15, 5]. The ROC (Receiver Operating Characteristic) curve represents the model performance by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) for different thresholds for classifying data points as anomalies based on their anomaly scores. The TPR is just another name for recall and is calculated the same as in Eq. (2.2). The FPR is the proportion of normal data that is falsely labeled as anomalous and is calculated as

$$FPR = \frac{\text{Number of falsely identified anomalies}}{\text{Total number of normal data}} \tag{2.4}$$

In Fig. 2.2 is an example ROC curve. The blue line represents a well working anomaly detection algorithm while the grey line is about what you would expect from random guesses. It is then quite common to summarize the information from a ROC curve into a single value by calculating the Area Under the Curve (AUC). A perfect model would have a ROC AUC score of 1, while random guesses would have one of 0.5. The blue algorithm in Fig. 2.2 has a score of 0.95.

It is important for an effective anomaly detection system to properly distinguish between anomalies and noise. Noise refers to random or irrelevant variations in the data that do not carry any meaningful information, which can arise from measurement errors, data collection artifacts or environmental factors. Anomalies and noise may appear very similar and make it difficult to distinguish between them since often the difference comes down to the subjective opinion of an analyst [3]. What constitutes an anomaly or noise is also quite different between

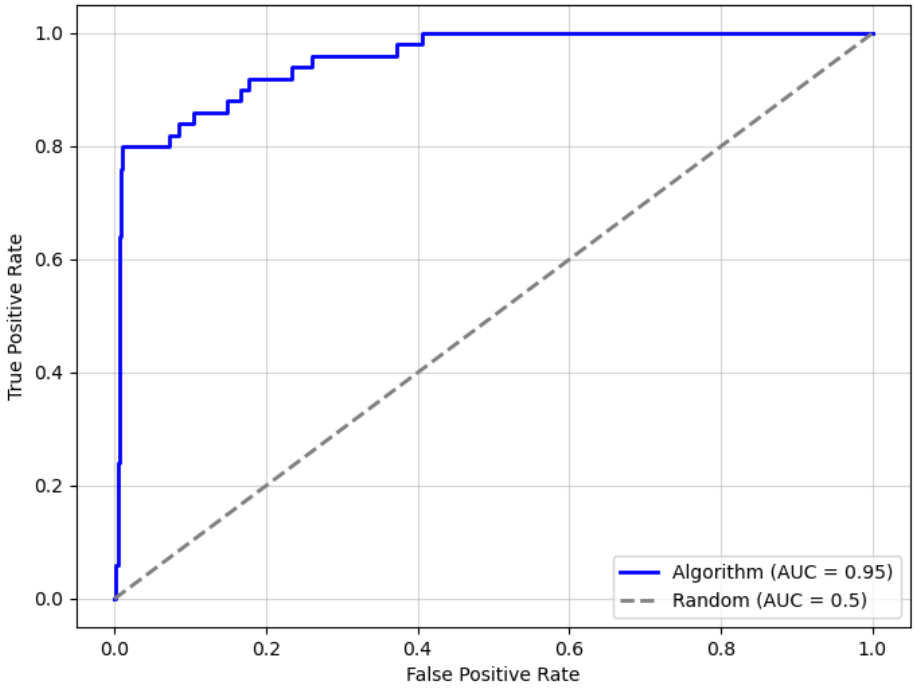

Figure 2.2: ROC Curve

different application domains. For example, a small change in the medical domain (e.g. a change in body temperature) can constitute an anomaly, but in the financial domain (e.g. small fluctuation in the stock market) can be viewed as normal [7]. Ways to combat noise are noise reduction beforehand or using robust detection methods such as in [41]. Another way might be to use appropriate thresholds that only looks at data points with a sufficiently high anomaly score.

Chapter 3

Generative Models

In modern machine learning, generative models have materialized as one of the main points of research. With them we have been able to generate realistic samples from text to images [6, 28, 14] and also understand and manipulate high-dimensional data [4, 20]. Generative models can be used in many different domains including creative arts, medical fields or anomaly detection [12, 29, 11, 31].

At its core generative models are algorithms that learn the underlying distribution of a given dataset. With this distribution a model is then able to generate new samples that resemble the original training data [14]. Again there are many different generative models that have been or are currently getting researched. The most used ones are Variational Autoencoders (VAE), which learn a representation of the data and generate new data by sampling from the latent space [20], and Generative Adversarial Networks (GANs), which consist of a generator and a discriminator that compete against each other to produce realistic data [14]. Another popular model, especially for generating image data, are diffusion models. They work by starting with random noise from a simple distribution and then iteratively denoising that noise to arrive at the wanted data [19].

Generative Models come with their own unique challenges. These can be specific to the model used or broadly apply to all kinds of different models. A specific issue of GANs for example is 'mode collapse', which results in the generator only creating a limited set of data patterns that does not capture the full diversity of the data distribution [23]. Another problem that some models have to deal with is their computational complexity and time required to generate a sample [16]. One challenge that every generative model has is the evaluation of the generated samples. Many different factors play into the evaluation and based on the data type there isn't an objective way to rate one model over another. In a lot of cases it comes down to a subjective visual inspection [16].

It can also be quite difficult to create a functioning generative model for high dimensional data. That comes down to the fact that high dimensional data often follows complex distributions, which require more training data and a lot more computational cost to properly learn.

There are many different applications of generative models. They can be used for data synthesis for generative text models, art or design [6, 14]. The generated data can then also be used to improve the performance of other machine learning tasks by letting them train on this synthetic data as well as on real data. That process is called data augmentation [36]. Generative models can also be used in connection with anomaly detection algorithms.

Chapter 4

Interplay between Generative Models and Anomaly Detection

As mentioned above there is a connection between generative models and anomaly detection. We want to first look at how this connection has been explored in related work and then expand on what we contribute to extend current research.

4.1 Related Work

Usually the connection between generative models and anomaly detection starts at generative models, which learn the data distribution and can generate normal samples [27]. If a sample cannot be generated it is likely that that sample is anomalous [33]. Similarly this can be viewed the other way around, where we search for a normal sample using the output of an anomaly detection algorithm, e.g. by training a model to minimize the overall loss of the model for normal data [41]. Our approach extends on this by also minimizing the loss of individual samples after a model has been trained.

It is viable for some generative models to directly be used for anomaly detection. One way is to use reconstruction-based models like Autoencoders or VAEs. As they learn to reconstruct the input data one can calculate the reconstruction error, which is the difference between the reconstruction and the original image. A high reconstruction error indicates anomalies since the models are only able to properly reconstruct normal data [32]. You could also directly analyze the latent space representation to detect anomalies for models that compress the data [41]. Another way to utilize generative models in anomaly detection tasks is by generating synthetic data that can then be used train anomaly detection algorithms. The main problem with training on anomalous data is that anomalies are rare, hence generating synthetic data can be beneficial. You can also use generative models to synthesise normal data that anomaly detection algorithms train on. This can lead to the detection algorithm better understanding normal data and better distinguishing it from anomalous data. The problem of evaluating synthetic data and their

generative models persists. Using it for anomaly detection brings an addition evaluation metric, how effective the generated data really is for anomaly detection in specific, i.e. is the synthetic data normal or are there generated samples that could be classified as as anomalies themselves.

4.2 Our Contribution

Existing studies primarily use anomaly detection to improve generative models, using anomaly scores as a loss function to align generated data with the normal distribution. We invert this approach by using anomaly detection to guide the generation of individual samples. By iteratively optimizing the anomaly score of a sample, we aim not only to generate samples that look normal but also to analyze whether different anomaly detection algorithms correctly learn what constitutes normal. This provides insight into each algorithms ability to learn the normal distribution while revealing potential limitations in the learned solution space.

Chapter 5

Methodology

In this chapter we want to explain the methods we used to generate images based on anomaly scores. We introduce the datasets used, go over the anomaly detection algorithms we evaluated and explain two different strategies for generating images. We also take a look at how we evaluate the generated images and in turn the models.

5.1 Datasets

For this work we used two different datasets. The first one we used is called MNIST [8]. It consists of grayscale images of handwritten numbers from 0 to 9. The images are of size 28×28. In total there are 70.000 images, 60.000 of which are in the training set and the other 10.000 are in the test set. Each of the 10 different numbers has roughly the same amount of images, both in the training set as well as in the test set. To use MNIST for anomaly detection, we defined one number as normal and every other number as anomalous.

The second dataset we used is called CIFAR-10 [22]. This dataset consists of 60.000 color images in 10 classes of different objects. The classes include objects like *automobile, cat, dog, ship* and *truck*. Of the 60.000 total images 50.000 are in the training set and 10.000 in the test set. Each class has exactly 5.000 training images and 1.000 test images. The images are of size 32×32, with additional color channels for a total size of $32 \times 32 \times 3$. Similar to MNIST, we treated one class as normal and the rest as anomalous.

5.2 Anomaly Detection Algorithms

In this work we used three different deep learning-based anomaly detection algorithms, namely Autoencoders, Deep SVDD and DEAN. We will explain the basics of each of the three as well as how each of their anomaly score is calculated.

5.2.1 Autoencoder

The first anomaly detection method we used is the Autoencoder (AE) [18]. AEs consist of a neural network that is trained to output a reconstruction of the input data. It does that by learning a compressed representation of the data in *latent space*. The neural network is made up of two different parts: the encoder and the decoder. The **encoder** compresses the input data into the latent representation while the **decoder** reconstructs the original data from the latent representation.

For our implementation of AEs we use convolutional layers [26], which use kernels that slide over the input to compute local patterns such as edges, textures or shapes. They help preserve spatial relationships between a local area of pixels. Our implementation uses the library TensorFlow [1].

AEs have many different use cases and have to be adapted to work for anomaly detection tasks [32]. To do this, we first define how we calculate the anomaly scores. These scores are also called the reconstruction error, as it quantifies the difference between the original input and the reconstructed output. To be more precise, for a input vector x consisting of n dimensions $x_1, x_2, ..., x_n$ and the reconstruction $\hat{x}$ produced by an AE, the reconstruction error is calculated as

$$Err_{recon}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{5.1}$$

We trained the AE only on normal data. This causes the reconstruction error (or the anomaly score) to be small for normal images, but higher for anomalous data, since the AE cannot reconstruct anomalous images as well as normal ones.

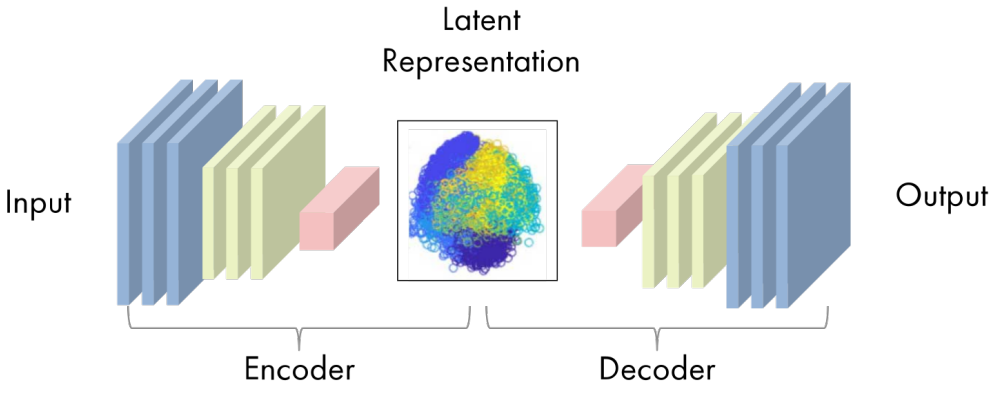

Figure 5.1: Abstract architecture of AEs. Figure by [38]

5.2.2 Deep SVDD

As the second anomaly detection method we used Deep SVDD [30], which is an extension of Support Vector Data Description [37]. Depp SVDD was primarily constructed for anomaly

detection tasks. The goal of Deep SVDD is to learn a hypersphere that encloses all the normal data. It does that by first mapping the data into a latent space. Then it tries to minimize the volume of the hypersphere while ensuring that all normal data falls within this sphere. Any data point that lies within the hypersphere is classified as normal data, while every data point that lies outside the hypersphere is classified as an anomaly. The anomaly score of a given point is calculated as the euclidean distance of the point in latent space to the center of the hypersphere. Fig. 5.2 shows a 2-dimensional example of how Deep SVDD functions. For our work we used the implementation of the library pyod [40].

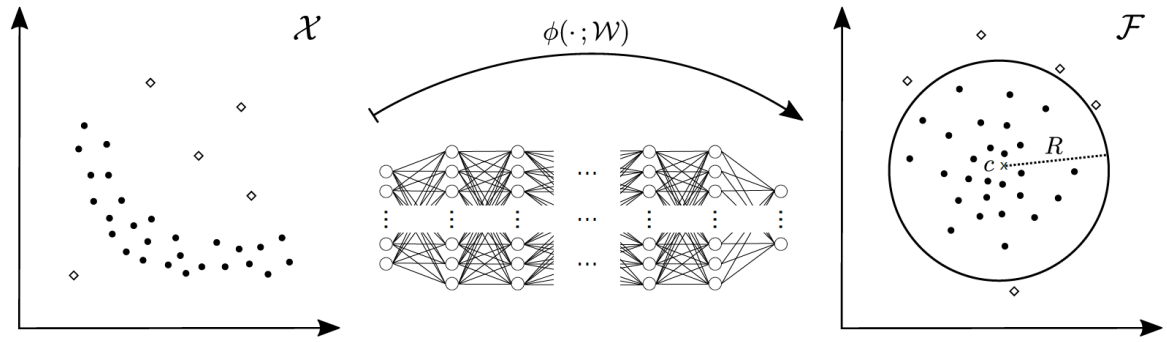

Figure 5.2: Deep SVDD learns a neural network transformation ϕ that maps the data from the input space $\mathcal{X}$ into the latent space $\mathcal{F}$. The hypersphere where normal data lies within and anomalies fall outside of is characterized by the center c and radius R. Figure by [30]

5.2.3 DEAN

Deep Ensemble Anomaly Detection (DEAN) [21] is an ensemble-based anomaly detection algorithm that combines multiple deep learning models to improve robustness and performance. To create a well performing ensemble it relies on four attributes:

- **Scalability:** Submodels need to be able to be generated quickly to make it possible to combine a lot of different models.
- **Depth:** Submodels need to be able to learn complex functions, thus needing each of them to be deep. Though the individual performance of each model is not as important as what the ensemble can learn from it.
- **Variability:** To benefit from having multiple different models their outputs need to be different from each other. This usually means the quality of each individual model has to not be optimal but good enough to properly distinguish between normal and anomalous data.
- **Consistency:** To combine multiple submodels, their outputs need to be in the same scale and to be comparable to each other.

DEAN fulfills these attributes by training the submodels with the same loss function and the same model setup but different initializations. To be more precise, DEAN uses *feature bagging*, meaning a model can only see *bag* many of the total features. Changing which features each submodel sees increases the variance between them and thus raises the quality of the ensemble. Since each submodel is trained on the same loss function, the resulting scores can easily be compared and combined. The anomaly score of the ensemble is calculated as the average of the output of all the submodels.

For this work we used the implementation of DEAN found in [21].

5.3 Generating Samples from Anomaly Scores

In this chapter we want to show what methods we used to generate images from the anomaly scores of the anomaly detection algorithms. Generating these images is an optimization problem where we want to minimize the anomaly score of a given input. For this we used two different methods, one where we directly used gradients from the underlying neural networks and one where we used an evolutionary algorithm. Both methods start with a random input and then modify this input to minimize the anomaly scores.

5.3.1 Gradient-based Method

When using neural networks, for each pass of data through the network gradients are calculated. These gradients can tell us what we have to change in the data in order to get a lower loss from the loss function. As our loss function is the anomaly score, the gradients can tell us how to change a image in order to minimize its anomaly score. This change depends on the learning rate and occurs gradually, so multiple iterations are required to properly minimize the anomaly score.

Using only these gradients can lead to the algorithm getting stuck in local minima. This is not necessarily a bad thing, as it can enable the generation of diverse normal samples (see for example Section 6.1.1). However it could also lead to the generated image being not a normal one, but instead being an anomaly. Using gradients is very efficient and provides fast convergence even for high-dimensional data. However, this method can only be used for anomaly detection algorithms that utilize neural networks and is also dependent on parameters like the learning rate and initialization.

5.3.2 Evolutionary Algorithm

Evolutionary Algorithms are inspired by the biological process of natural selection, in which a population of a species evolves over time to adapt to their environment. These types of algorithms have been widely studied in the field of artificial intelligence and optimization [10]. In a population there is a fixed number of individuals. In our work, the individuals are images.

In each generation, the individuals get ranked using a fitness score, the anomaly score in our case. Then, a new population for the next generation is formed through a combination of three possible processes:

- **Mutation:** Randomly modifies an individual to search in new regions of the search space.
- **Crossover:** Combines two individuals (*parents*) to create a new one (*child/ offspring*), which hopefully captures the strengths of both parents.
- **Selection:** Selects the best performing individuals based on their fitness score.

In our work we produce a new population by combining two images by taking the average pixel value between them and sometimes mutating them by adding small values to each pixel. These small values are generated at random by a normal distribution with a mean of 0 and a standard deviation of 0.01. Mutations happen 50% of the time. Images with a lower anomaly score have a higher likelihood of getting chosen as a parent. This repeats for a fixed number of generations.

Evolutionary algorithms are robust against local minima, since random mutations can lead to worse results in the short term which eventually lead to a overall better result. They are also very universal and can be fairy easily adapted to different types of anomaly detection algorithms or even completely different optimization tasks. However they are computationally expensive, since they require the fitness scores of a large population to be evaluated over a large number of generations. To add to that, evolutionary algorithms require the fine-tuning of several different parameters like the population size, mutation rate, and more.

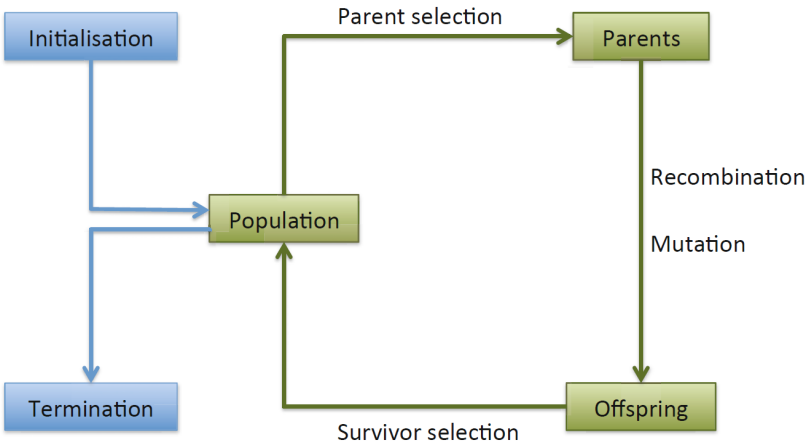

Figure 5.3: The general scheme of an evolutionary algorithm as a flowchart. Recombination represents Crossover. Figure from [10]

5.4 Evaluation Method

In order to compare different anomaly detection algorithms, we need evaluation metrics. For our work, we generate images and evaluate them via a **visual inspection**, determining whether they appear normal or anomalous. This method is simple and intuitive, especially for image data, but has several limitations, like its subjectivity and lack of quantitative measures.

While quantitative metrics could provide objective scores, they cannot guarantee that an image looks 'right' to humans; a generated image might score well numerically but still appear anomalous to human observers. Given the scope of our work and the straightforward nature of the datasets used, a visual inspection offers a practical and effective way to evaluate the performance of the anomaly detection algorithms based on their generated images.

Chapter 6

Experimental results

In this chapter we experimentally evaluate the performance of the different anomaly detection algorithms introduced in Section 5.2. For that we will generate images by minimizing the anomaly score of an individual sample using the methods from Section 5.3.

6.1 Autoencoder

6.1.1 Gradient-based minimization

First we generate images using the gradient-based optimization on the Autoencoder. Since this method converges fast, the images are generated with 500 iterations. We start the generation with randomly generated noise and then iterate on that using a learning rate of 0.01. For the generated images in this as well as in all other chapters, the anomaly score of an image is given as the title of that image.

Fig. 6.1 shows the generated MNIST images for the Autoencoder only being trained on the normal class *3* using gradient-based minimization. The generated images appear to be well constructed images and show no clear anomalies. Similar results can be seen for the class *4* in Fig. 6.2. Here, the generated images show more variation between them, which is also the case in the training set. These results suggest that Autoencoders are capable of properly learning the normal distribution of the MNIST dataset. It also appears that it can learn the full distribution and not only small parts of it by generating a variety of normal-looking images.

Using the CIFAR-10 dataset, the Autoencoder seems not to be able to properly learn the normal distribution. Fig. 6.3 shows generated CIFAR-10 images for the normal class *frog*. The images show a green and brown shade while not having any discernible patterns that could indicate a normal image. Similar results can be seen for the normal class *airplane* in Fig. 6.4. Again, the images show no normal patterns. This time however the images are showing a blue and brown shade. The difference in color could be an indication that at least some characteristics of the normal class is learned, since most frogs are green or brown and usually are on grass or dirt while the class *airplane* often captures the blue sky.

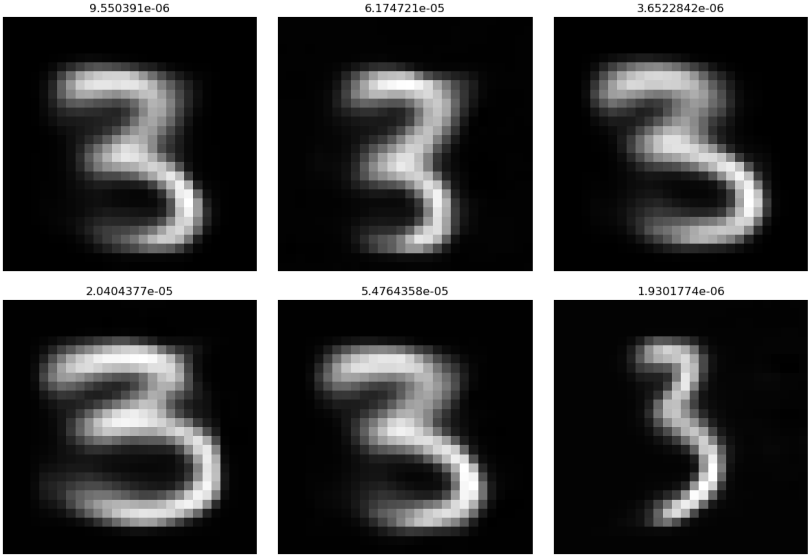

Figure 6.1: Generated MNIST images of class *3* using gradients of an Autoencoder. Numbers above the images represent the anomaly score of the generated image

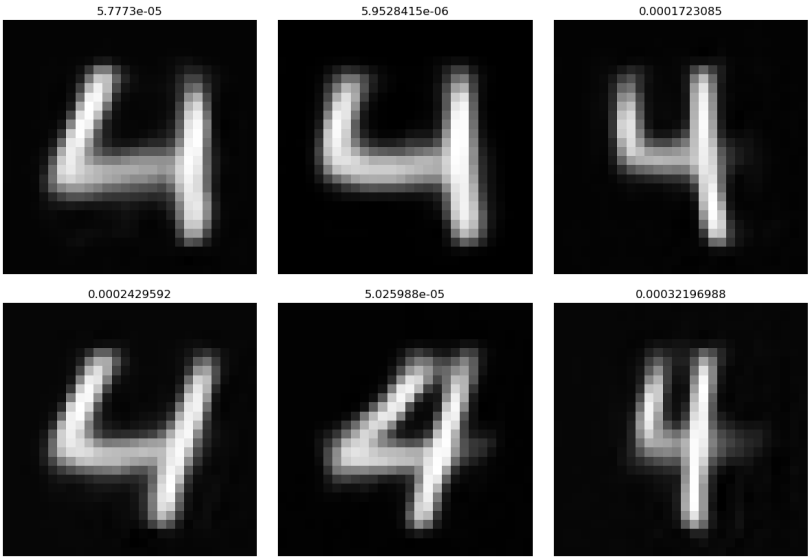

Figure 6.2: Generated MNIST images of class *4* using gradients of an Autoencoder

6.1.2 Minimization through Evolutionary Algorithms

Next we generated images using a evolutionary algorithm. Fig. 6.5 shows generated MNIST images of class *3* (top) and class *4* (bottom) while Fig. 6.6 shows generated CIFAR-10 images of class *frog* (top) and class *airplane* (bottom). The MNIST images are very similar to the ones generated using gradients, showing a variety of different normal images with no detectable anomalies. However the CIFAR-10 images show even less structure than the ones generated using gradients. Both images on the left show a very slight shade of green or blue respectively, while both images on the right show no difference between them amounting to random noise.

Figure 6.3: Generated CIFAR-10 images of class *frog* using gradients of an Autoencoder

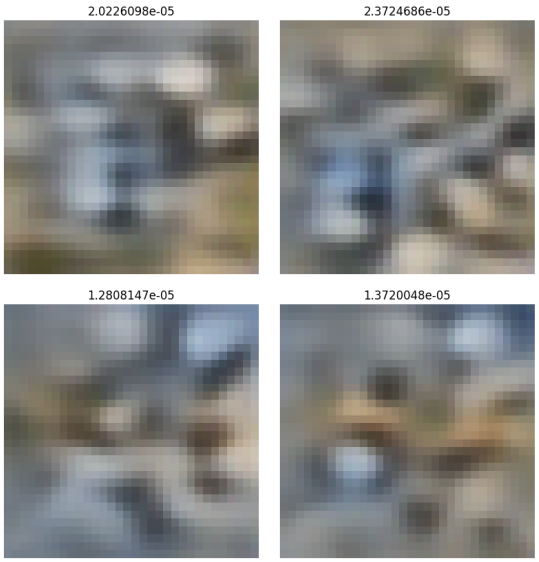

Figure 6.4: Generated CIFAR-10 images of class *airplane* using gradients of an Autoencoder

Both methods of generating images show normal-looking samples for the MNIST dataset while generating abnormal-looking samples for the CIFAR-10 dataset, with the evolutionary algorithm generating more anomalous-looking samples.

Due to these results and the fact that evolutionary algorithms are more universal and can be adapted to fit various anomaly detection algorithms very quickly, from here on we will only generate images using an evolutionary approach. We will be using an evolutionary algorithm with a population size of 150 images and a mutation rate of 0.5 with a different number of generations for different algorithms. Increasing the population size leads to a faster convergence to a small anomaly score but comes with higher computation time. Increasing it to over $\sim$100 leads to

diminishing returns, so we opt for the largest population size where the added computation time is not noticeable - a population size of 150. The mutation rate has a similar impact, where a lower mutation rate leads to slower convergence to a small anomaly score and a high one to faster convergence. To add to that, decreasing it too close to 0 will lead to practically no change to the best starting image, whereas increasing it too close to 1 will lead to the generated image having a higher anomaly score than what is possible with a lower mutation rate. We felt that a mutation rate of 0.5 is a good middle ground to generate the best possible images in a reasonable amount of time. The mutation rate goes hand in hand with the noise generated that gets added to the images. Changing the value of the random noise that gets added has a very similar, but more extreme effect than changing the mutation rate. We will be using random noise generated by a normal distribution with a mean of 0 and a standard deviation of 0.01. With those parameters, the evolutionary optimization was able to generate normal-looking images after as little as 1.000 generations, though the best looking ones took around 4.000-5.000 generations. More generations never decreased the quality of the generated images, though at a certain point increasing the number of generations had no effect on both the quality as well as the anomaly score of the generated image. The number of generations varies between the algorithms we use, primarily due to the computational time required. For Figs. 6.5 and 6.6 we used 5.000 generations.

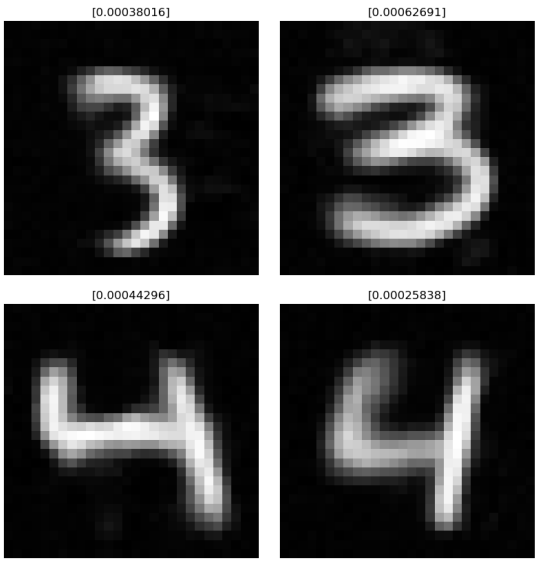

Figure 6.5: Generated MNIST images of class *3* (top) and class *4* (bottom) using an evolutionary algorithm on an Autoencoder

Figure 6.6: Generated CIFAR-10 images of class *frog* (top) and class *airplane* (bottom) using an evolutionary algorithm on an Autoencoder

6.2 Deep SVDD

6.2.1 Minimization through Evolutionary Algorithms

As explained above we will only use the evolutionary algorithm to generate images and do not generate images with Deep SVDD using the gradient-based method. We use the implementation of Deep SVDD from the pyod library [40] with no hyperparameter optimization. The generated images with Deep SVDD for the MNIST dataset can be found in Fig. 6.7 for class *3* and in Fig. 6.8 for class *4*, while the generated CIFAR-10 images of class *frog* can be found in Fig. 6.9 and for class *airplane* in Fig. 6.10. They were all generated with 30.000 generations.

All the generated images with Deep SVDD appear anomalous. There are no patterns visible and look much like random noise. The generated MNIST images for class *3* as well as for class *4* have a noticeable contrast in pixel variance between the central and edge regions. In the middle of the images, there is a higher variance between pixels next to each other, resulting in a less homogeneous space with sharp transitions and hard cuts. In contrast, the edges of the images show a lower variance between neighboring pixels, leading to a smoother, more homogeneous 'landscape of noise' with more gradual transitions and fewer abrupt changes. This however has no real impact on if the images look normal or anomalous since both the central as well as the outer region both contain only noise and no normal patterns. But it could be an indication that Deep SVDD learns part of the normal distribution, since in the normal images there is a lot more pixel variance in the center due to the digits and much less variance around the edges since it is a solid black in all the normal images. The generated CIFAR-10 images have no difference between each other or between regions in the image. They all look very anomalous, being made up of 'consistent' random noise and appear even more anomalous than the MNIST ones.

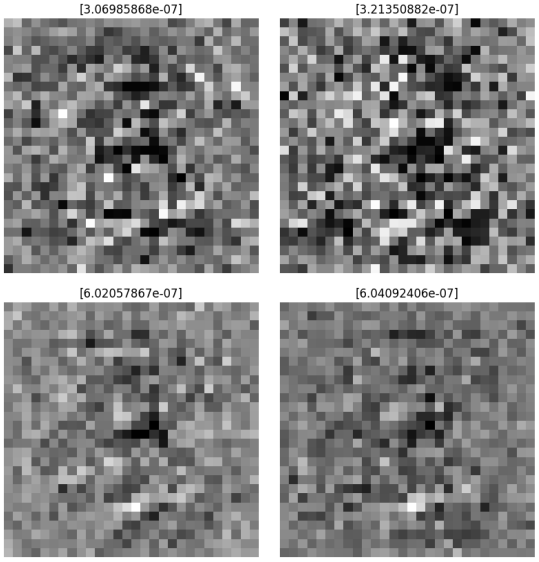

Figure 6.7: Generated MNIST images of class *3* on Deep SVDD

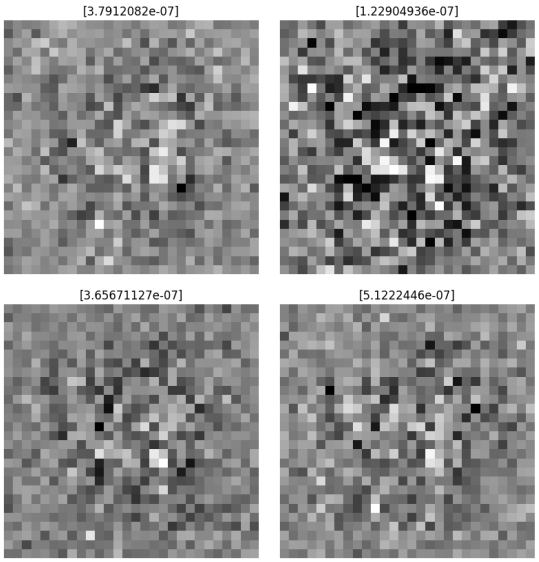

Figure 6.8: Generated MNIST images of class *4* on Deep SVDD

6.2.2 $k\%$ Fixed Pixels

Since Deep SVDD was not able to generate normal images by itself, we wanted to see if guiding the generation with normal patterns would help generate normal images. For this we fixed $k\%$ of the generating image to pixels of an already normal image. The evolutionary algorithm only minimized the other $1 - k\%$ fraction of remaining pixels. The results of that experiment can be found in Fig. 6.11, generated using the evolutionary algorithm with 30.000 generations. For each generated image we also show which pixels are fixed to the left of it. A white pixel in the "Fixed Pixels" image means the pixel is fixed to one of a normal image, while the black ones

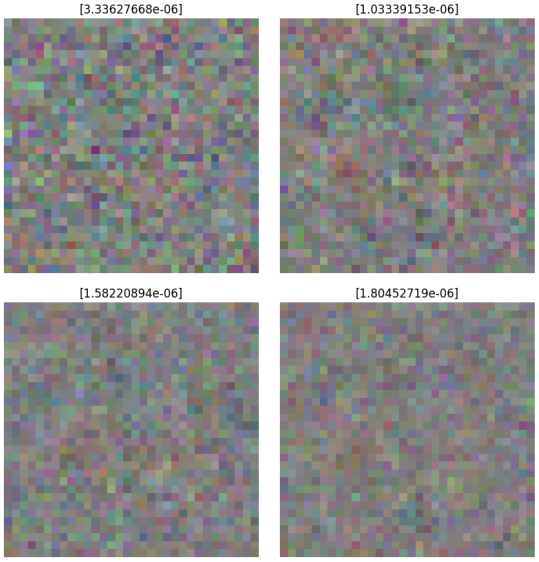

Figure 6.9: Generated CIFAR-10 images of class *frog* on Deep SVDD

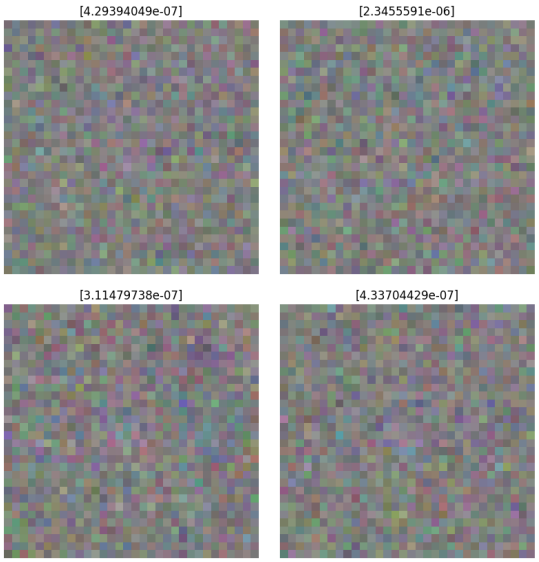

Figure 6.10: Generated CIFAR-10 images of class *airplane* on Deep SVDD

are subject to minimization. When looking at the generated image with 95% of the pixels fixed, Deep SVDD was still unable to adjust the other 5% to match the normal image. The pixels that are not fixed are clearly visible in every image. Even the anomaly scores for the images with 30% and 95% fixed pixels is very similar, even though visually the 30% image looks very anomalous while the 95% image looks practically normal. This leads us to conclude that Deep SVDD is not able to properly learn the normal distribution of the MNIST dataset or even parts of it. Instead, the distribution Deep SVDD learned classifies a very anomalous-looking image and a practically normal-looking one with the same anomaly score. Generating normal images seems not viable even with significant guidance.

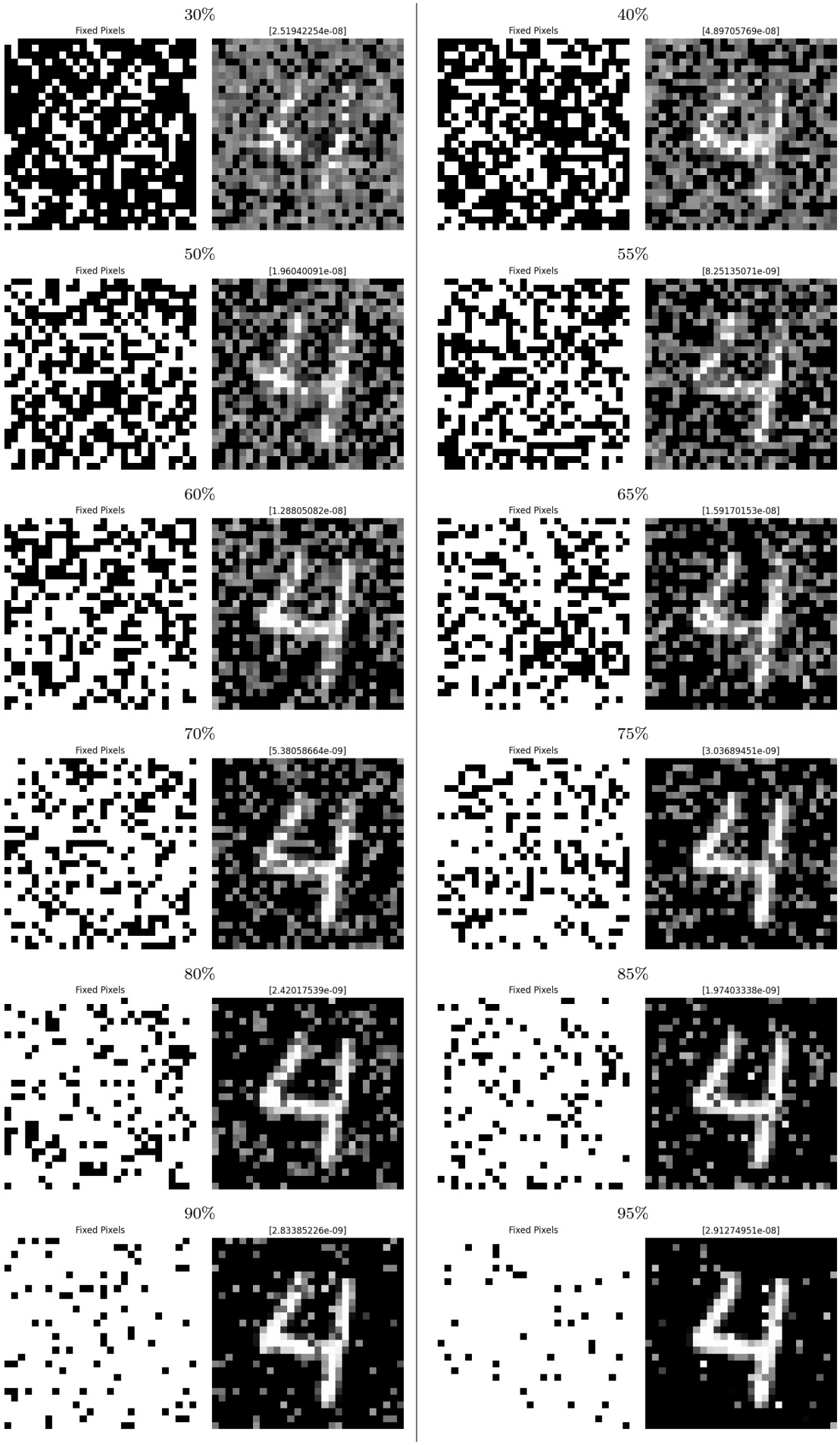

Figure 6.11: Generated MNIST images of class *4* on Deep SVDD with $k\%$ fixed pixels

6.2.3 Initial Population with normal images

We also used an alternative method of guiding the generation towards normal-looking images. So far, the initial population of the evolutionary algorithm was comprised of randomly generated images. Instead, we help the generation by now having only half of the initial population made up of randomly generated images while the other half is made up of normal samples from the test set. Fig. 6.12 shows these generated images after 1.000, 15.000, 30.000, 100.000 and 250.000 generations for class *4* of the MNIST dataset.

In the short term for a smaller number of generations this help seems to work, as the images appear quite normal, having only a slight bit of noise in the background while the '4' in the foreground can be clearly identified as such. Though the '4' appears more like a blurry average of all possible normal images than one specific one. After more generations the generated images devolve into looking more and more abnormal, eventually ending in random noise after 250.000 generations. Interesting to note is also the fact that the anomaly score of the generated images decreases with a higher number of generations, meaning the algorithms judges a normal-looking image as more of an anomaly than random noise. In fact, the algorithm judges the generated images after 100.000 and after 250.000 generations as normal. This supports our conclusion that Deep SVDD is not able to properly learn any parts of the normal data distribution.

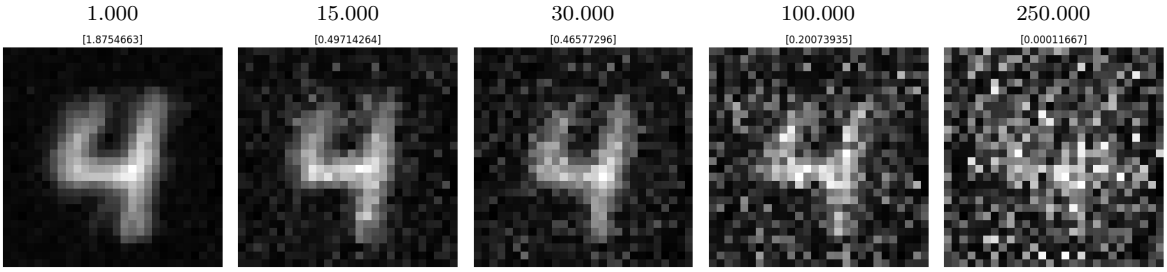

Figure 6.12: Generated MNIST images of class *4* on Deep SVDD with half of the initial population consisting of normal images from the test set after different number of generations

6.3 DEAN

6.3.1 Minimization through Evolutionary Algorithms

As the third algorithm we generated images with DEAN. As before, we will be using only the evolutionary algorithm to optimize the anomaly scores. We will be using the implementation found in [21]. The experiments in this initial section use default values for all parameters, including an ensemble size of 100 submodels. In Section 6.3.2 we optimize this hyperparameter, with its resulting optimized value then adopted for all subsequent experiments. Due to the high computational cost required to generate images with DEAN we will generate images with only 1.000 generations in this chapter.

Fig. 6.13 shows generated images using DEAN. The top images are of class *4* from the MNIST dataset while the bottom ones are of class *frog* of the CIFAR-10 dataset. The generated images from both datasets appear very anomalous and are akin to random noise. These results suggest that DEAN, similar to Deep SVDD, may not be able to fully capture the normal data distribution. To validate this hypothesis, we will first optimize the number of submodels hyperparameter in DEAN, then guide the generation of images with the same two methods used for Deep SVDD.

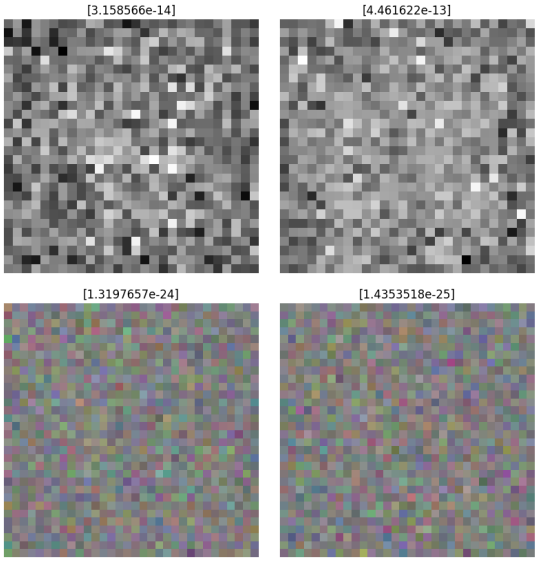

Figure 6.13: Generated MNIST images of class *4* (top) and CIFAR-10 images of class *frog* (bottom) on DEAN

6.3.2 Performance with different number of submodels

Since DEAN is an ensemble based anomaly detection algorithm one the hyperparameters you can optimize is the number of submodels in the ensemble. The images in Fig. 6.13 were generated with 100 submodels. We want to determine whether increasing (or decreasing) the number of submodels has any significant impact on the quality of the generated image. Fig. 6.14 shows generated MNIST images using DEAN with 30, 100, 250, 500, 100, 1500 and 1750 submodels. As mentioned above, the images were generated with 1.000 generations. With more models the outer region of the images appear to become darker. However each generated image, regardless of the number of submodels, shows a very anomalous-looking image akin to random noise. The fact that in normal images the outer regions always are a solid black suggests that DEAN models with a higher number of submodels may better learn the normal data distribution, as evidenced by the outer regions getting darker with more submodels. It would be interesting to expand this experiment with even more submodels, however due to software limitations and the time required we were only able to generate images with up to ∼1750 submodels. In the following we will generate images with DEAN using 1500 submodels.

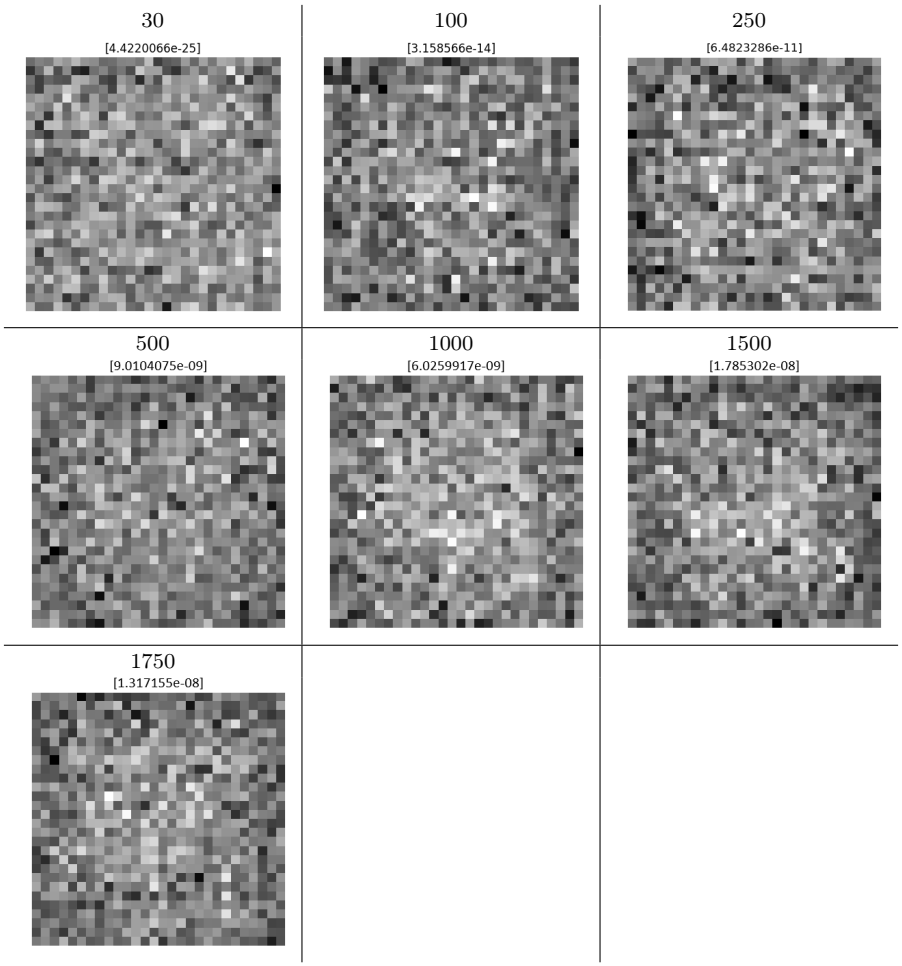

Figure 6.14: Generated MNIST images of class *4* on dean with different number of submodels

6.3.3 $k\%$ Fixed Pixels

Next we tried guiding DEAN towards generating normal images by fixing $k\%$ of pixels to those of a normal image. The results can be found in Fig. 6.15. Again, the images are generated with 1.000 generations. Compared to Deep SVDD, DEAN is able to produce images that appear much more normal. Starting at about 85% fixed pixels the generated images look very close to being normal, only having very few pixels that are significant anomalies. At 50% fixed pixels the shape of the '4' in the center is clearly visible. Comparing to Deep SVDD this shape becomes visible later, at about 60% fixed pixels. These results suggest that DEAN is not able to properly learn the normal distribution of the MNIST dataset but instead only learns parts of it. Generating normal images seems possible with some guidance. DEAN appears to better learn the normal data distribution than Deep SVDD.

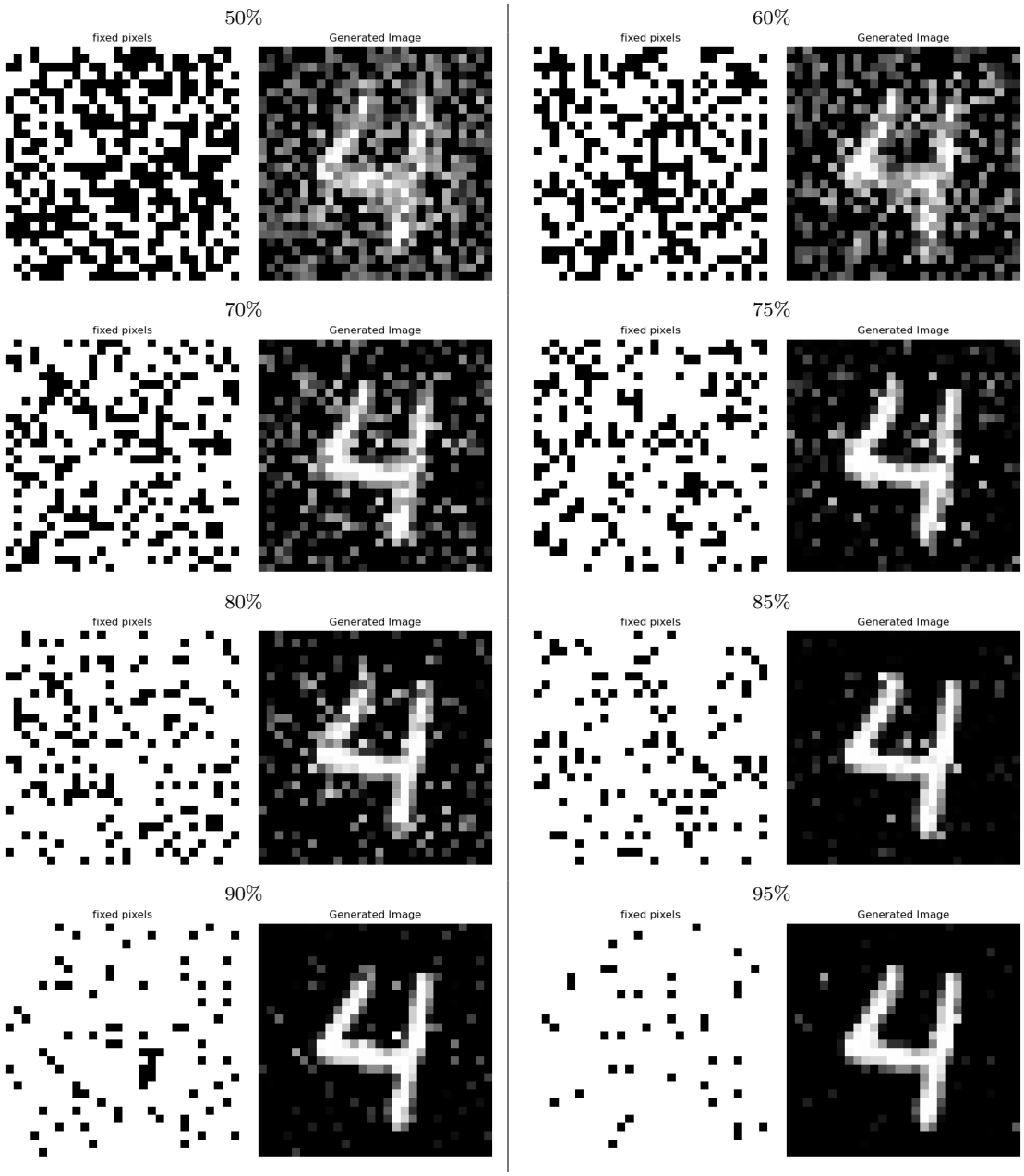

Figure 6.15: Generated MNIST images of class *4* on dean with $k\%$ fixed pixels

6.3.4 Initial Population with normal images

Finally we guide the generation towards normal-looking images by having half of the initial population be comprised of normal images from the test set. The generated images after 1.000 and 15.000 generations are shown in Fig. 6.16. The generated images appear normal, while there is noise along the edges that is more noticeable for 15.000 generations than for 1.000 generations. This suggests that DEAN may, similar to Deep SVDD, converge towards a image looking more like random noise than a normal image. However, due to the computational cost, we were unable to generate images with more generations. Compared to Deep SVDD, the images

DEAN generated overall look more normal, with the '4' in the center appearing more as a specific instance of a normal image rather than a blurry average of possibilities, and overall less noise. These results support the conclusion that DEAN can better capture the normal data distribution compared to Deep SVDD.

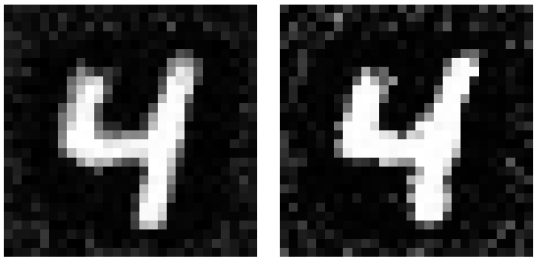

Figure 6.16: Generated MNIST images of class *4* after 1.000 generations (left) and after 15.000 generations (right) on DEAN with half of the initial population consisting of normal images from the test set

6.4 Other Anomaly Detection Algorithms

We evaluated other anomaly detection algorithms as well, namely Copula Based Outlier Detector (COPOD) [24], Histogram-based Outlier Detection (HBOS) [13], Isolation Forest (IForest) [25], Gaussian Mixture Model (GMM) [2], Clustering Based Local Outlier Factor (CBLOF) [17] and One-class Support Vector Machines (OCSVM) [9, 34]. For this we generate multiple MNIST images of class *4* with each of the algorithms. Like with Deep SVDD and DEAN, we generated images with different fractions k% of the pixels fixed to ones of a normal image. We started with 0% fixed pixels and increased that by 10% each image. The algorithms are all implemented using the pyod library [40] with default values for the parameters without hyperparameter optimization. The results of this experiment can be found in Fig. 6.17 with Autencoder, Deep SVDD and DEAN included for completeness. Even more generated images can be found in Appendix A. All of the images were generated with 30.000 generations, apart from those using COPOD (5.000 generations), Autoencoder (5.000 generations) and DEAN (1.000 generations).

The generated images from CBLOF and OCSVM show that these two algorithms can generate normal images from completely random noise, indicating that they can properly learn the normal distribution. However in contrast to Autoencoders, they do not capture the full diversity of it. CBLOF captures only a small part of the distribution and will generate the same normal image even with different random starting noise. After re-training the model the image might look different, but will again be the same one for that trained model for different starting noise, as shown in Fig. 6.18. OCSVM however generates a fairly blurry normal image that is consistent for re-training, shown in Fig. 6.19.

HBOS and IForest seem comparable to Deep SVDD, not able to properly capture the normal distribution and thus not able to generate a normal looking image even for large k.

COPOD and GMM are similar to each other, both having the center full of noise with the edges being black. With COPOD this effect is stronger (i.e. the center noise is smaller) than with GMM. This leads to the generated images looking more normal than from HBOS, IForest or Deep SVDD. This indicates that COPOD and GMM both capture a small part of the normal distribution (i.e. the outer regions of the image) but not being able to capture the majority of it (i.e. the center of the image).

Overall, the evaluated algorithms can be grouped into three categories reflecting their ability to generate normal-looking images:

- **Algorithms that can generate normal-looking images without guidance:** This group includes Autoencoder, OCSVM, and CBLOF. Their performance suggests the ability to properly learn the normal data distribution.
- **Algorithms that require guidance to generate normal-looking images:** COPOD, GMM, and DEAN fall into this category. While they capture certain aspects of the normal distribution, their reliance on guidance indicates incomplete learning of the normal data distribution.
- **Algorithms that struggle to generate normal-looking images even with guidance:** HBOS, Deep SVDD, and IForest exhibit this behavior, implying fundamental limitations in learning the normal data distribution.

For each algorithm we have included the AUC ROC score. It appears as though there could be a correlation between the ROC AUC score and the quality of the generated images, with higher ROC AUC scores leading to more normal looking images. That is supported by the fact that 3 of the best 4 performing algorithms can generate normal images without guidance and the least normal images being generated by some of the the worst performing algorithms. However, DEAN with the highest ROC AUC score is not able to generate normal looking images without guidance. On top of that COPOD and GMM generate images of similar quality while having vastly different ROC AUC scores. Proving a correlation requires significantly more research than what is conducted in this thesis.

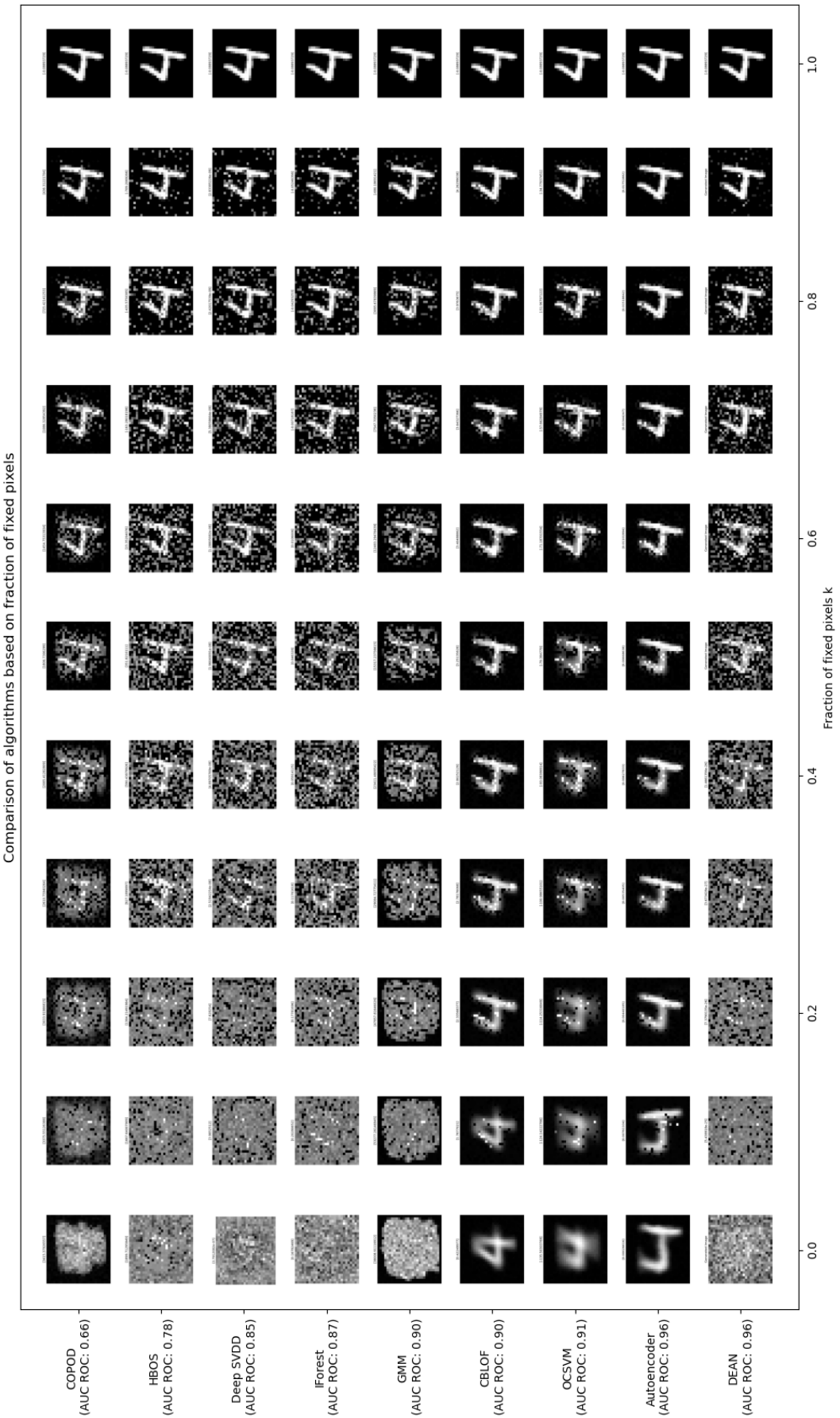

Figure 6.17: Generated MNIST images of class *4* on various anomaly detection algorithms with $k\%$ fixed pixels

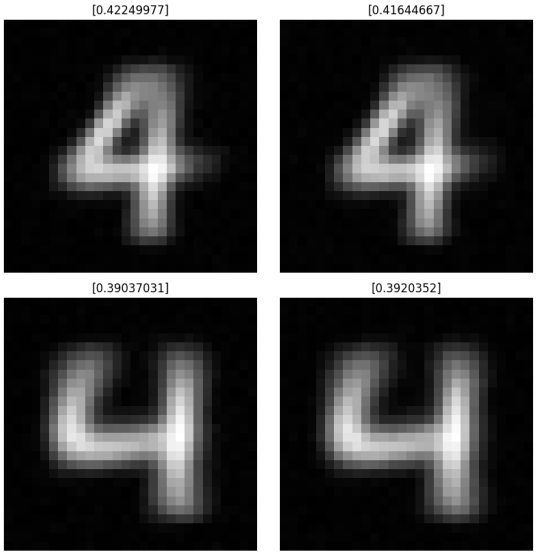

Figure 6.18: Generated MNIST images of class *4* on CBLOF. Model got re-trained after generating top images

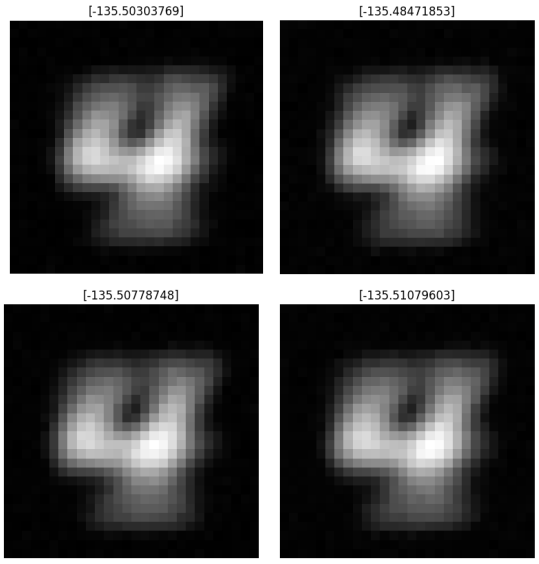

Figure 6.19: Generated MNIST images of class *4* on OCSVM. Model got re-trained after generating top images

Chapter 7

Conclusion and Future Work

In this thesis, we evaluated the performance of anomaly detection algorithms by generating images from anomaly scores. We focused our experiment on three deep learning-based algorithms: Autoencoder, Deep SVDD and DEAN. We used both a gradient-based a evolutionary method to generate images and assessed their quality through a visual inspection.

We found that Autoencoders can generate normal-looking images for the MNIST dataset, indicating that they can learn the normal data distribution effectively. However, they struggle with the more complex CIFAR-10 dataset. Deep SVDD and DEAN struggle to generate normal-looking images for both datasets while still being able to effectively detect anomalies. This suggests that they only learn parts of the normal distribution. Guiding the generation process by fixing a percentage of pixels ($k\%$) improves the quality of the generated images, particularly for DEAN. However, Deep SVDD remains unable to generate normal-looking images even with significant guidance. We also evaluated other anomaly detection algorithms on the MNIST dataset. We found that CBLOF and OCSVM are able to generate normal-looking images without guidance, but their ability to capture the full diversity of the normal distribution is limited. COPOD and GMM were able to capture parts of the normal distribution, being able to generate normal-looking images with some guidance, while HBOS and IForest were unable to generate normal-looking images even with significant guidance.

Our approach of generating samples from anomaly scores provides a new way to evaluate and interpret the behaviour of anomaly detection algorithms, which could lead to the development of more robust and interpretable models in the future. Our findings highlight the importance of evaluating not only the anomaly detection performance but also their ability to learn the underlying distribution of the normal data. This is particularly relevant where understanding the normal distribution is critical, such as in medical image processing.

While our approach provides valuable insights into the performance of anomaly detection algorithms, it has several limitations. Our evaluation relies on a subjective visual inspection, which could be expanded upon in future work by incorporating quantitative measures to assess the quality of generated images. Our experiments also were limited to only two datasets, which

may not fully represent the complexity of real-world data. To add to that, we only evaluated a limited number of anomaly detection algorithms. Future work could build upon our work by using additional or more complex datasets to provide a more comprehensive evaluation of the algorithms, as well as using more anomaly detection methods. This could also help in determining whether there is a correlation between the AUC ROC score of an algorithm and its ability to generate normal-looking samples. Future work that applies our approach to real-world applications, like fraud detection or medical diagnoses, would demonstrate its practical utility and provide insights into its performance in real-world scenarios.

In conclusion, this thesis provides a novel approach to evaluating anomaly detection algorithms by generating samples from anomaly scores. Our findings show the strengths and weaknesses of current algorithms and offer valuable insight into their ability to learn the normal data distribution. We hope that this work inspires future work that ultimately leads to better performance of anomaly detection and data generation in real-world applications.

Appendix A

Generated Images by other Anomaly Detection Algorithms

All generated images in the following are generated images of class *4* of the MNIST dataset. The images are generated using an evolutionary algorithm with $k\%$ of the pixels fixed to those of a normal image. The images were generated with 30.000 generations, except for COPOD, which used 5.000 generations. White pixels in the "Fixed Pixels" images are fixed, black ones are subject to minimization by the evolutionary algorithm.

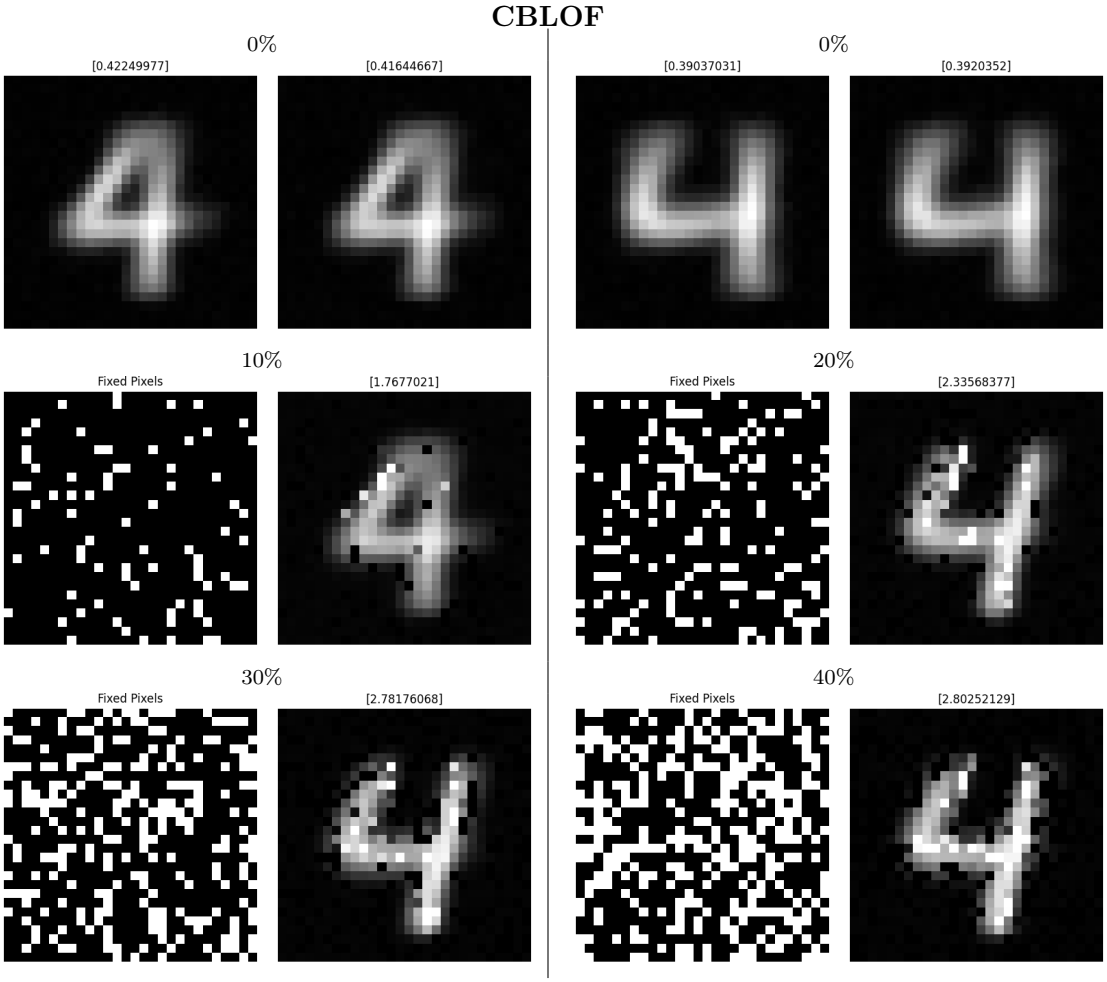

Figure A.1: Generated images on CBLOF. Top four 0% images are all generated images and have no fixed pixels. More images in Fig. A.2

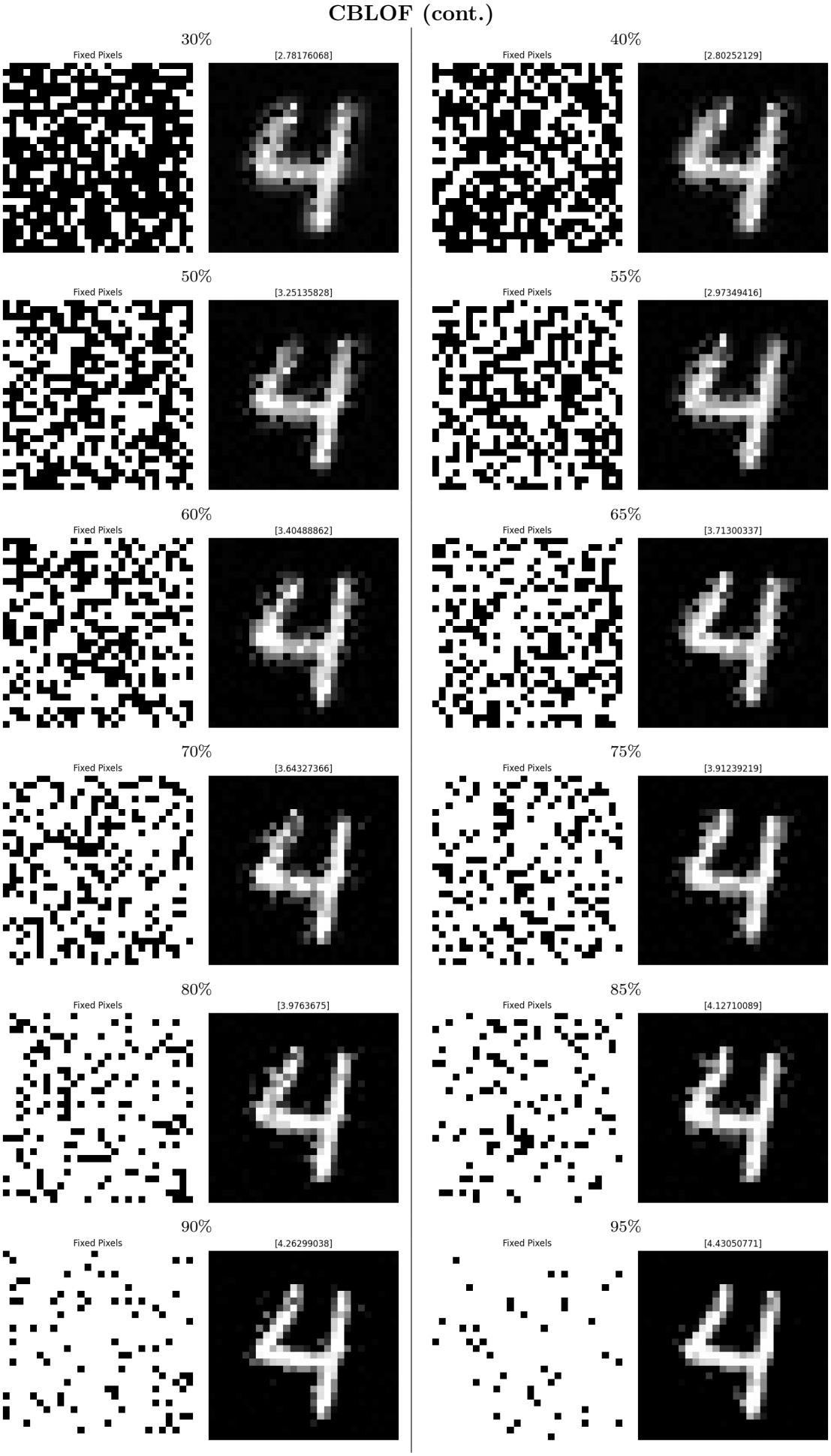

Figure A.2: Generated images on CBLOF (cont.)

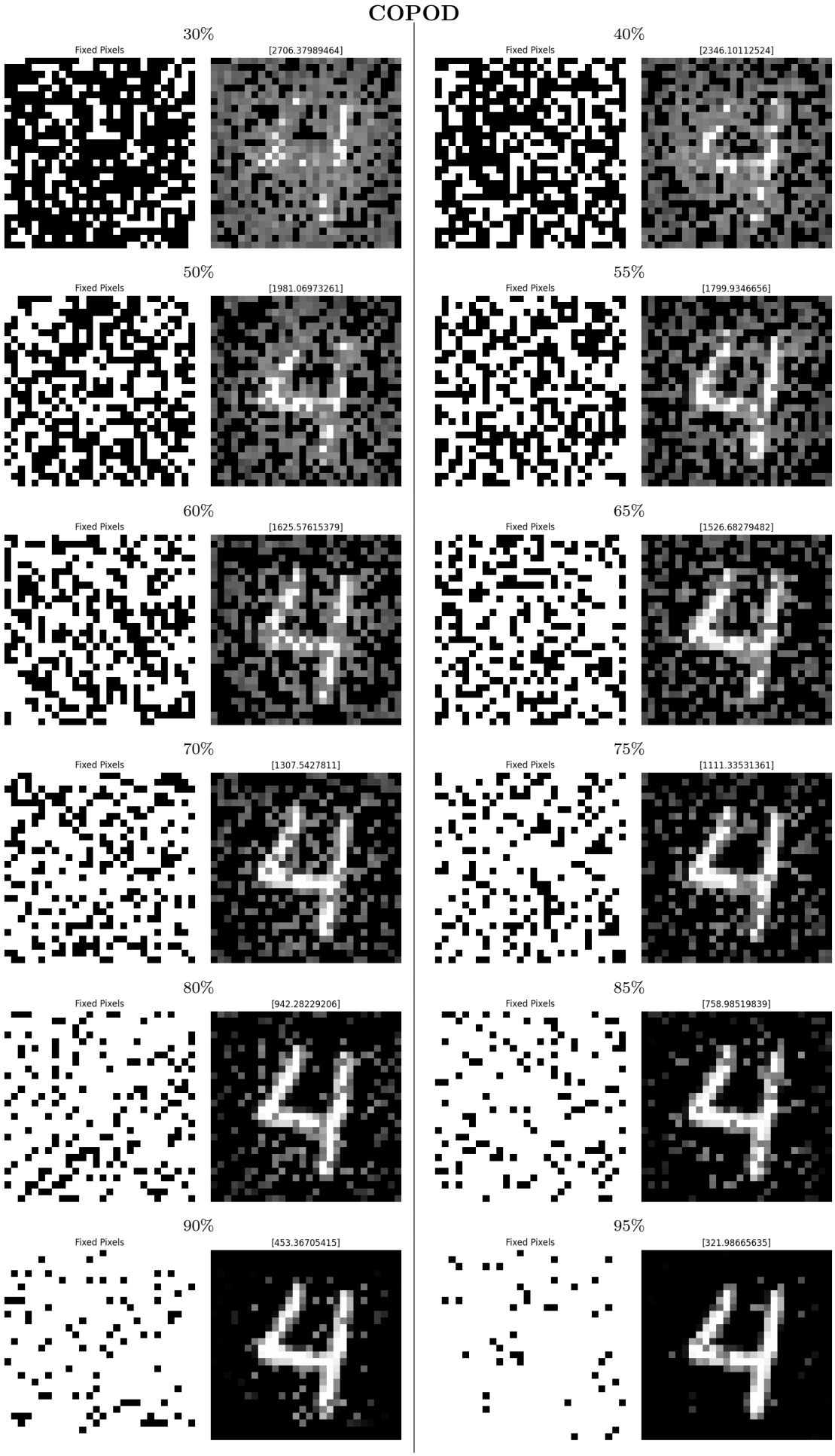

Figure A.3: Generated images on COPOD

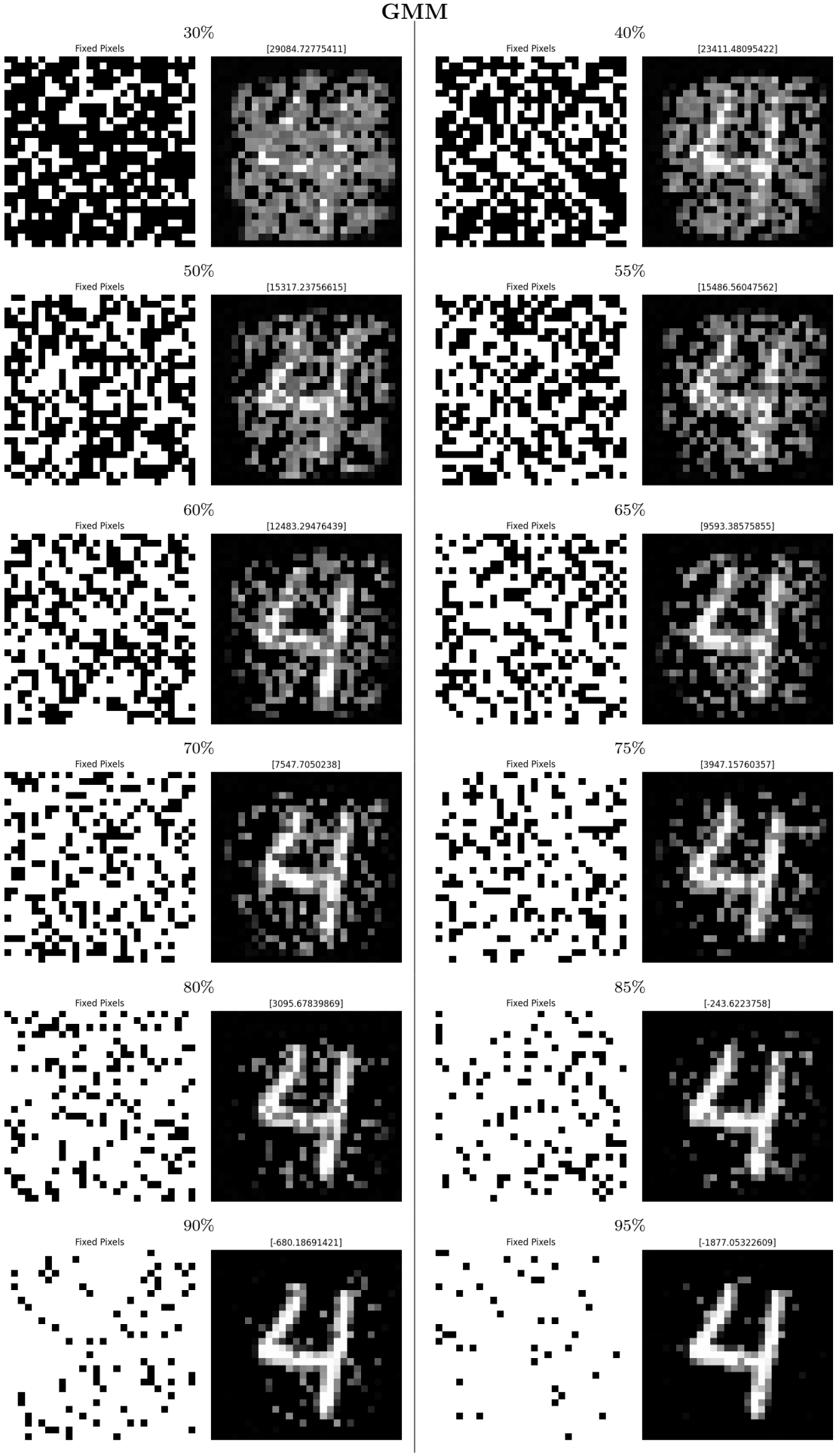

Figure A.4: Generated images on GMM

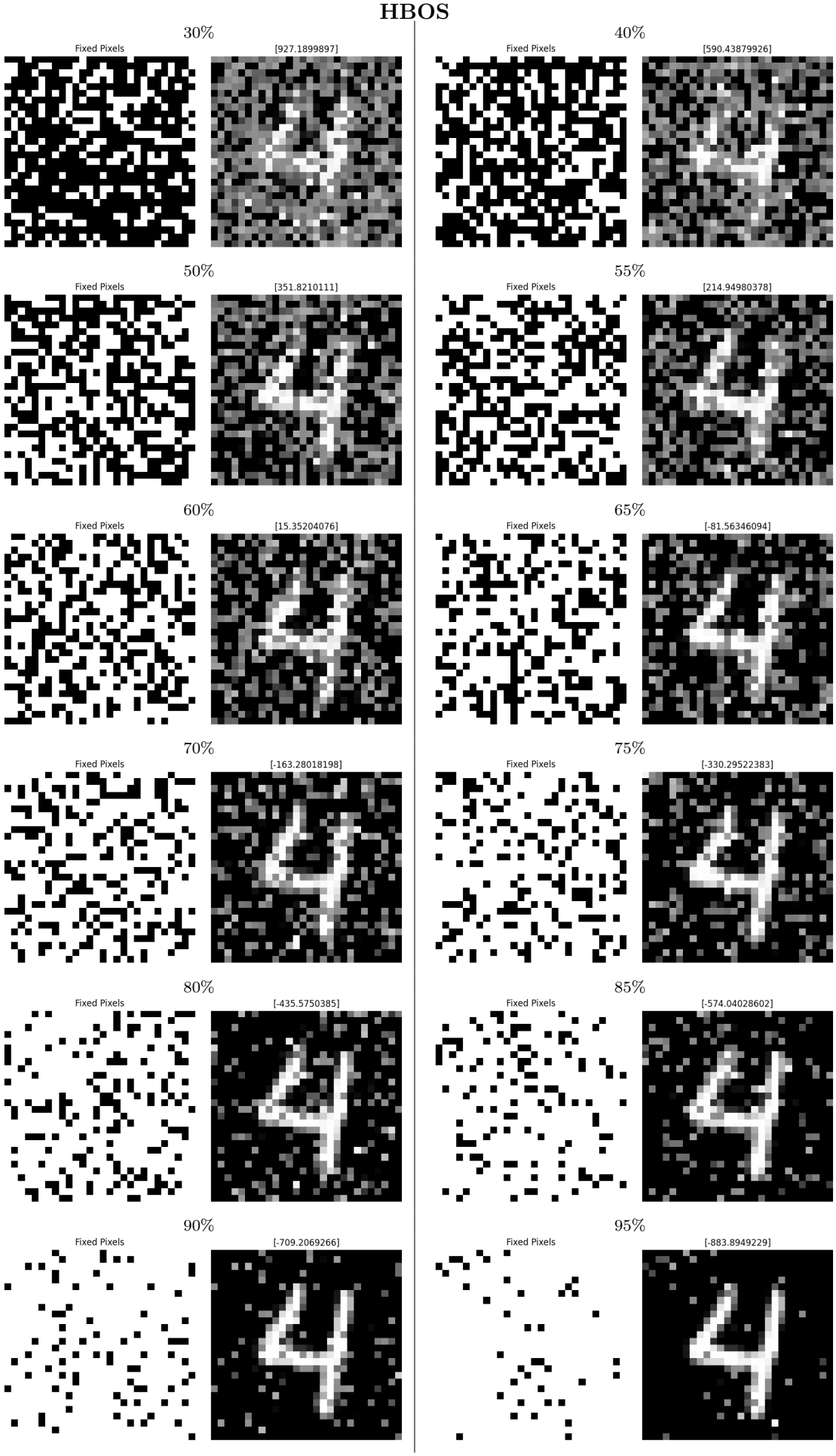

Figure A.5: Generated images on HBOS

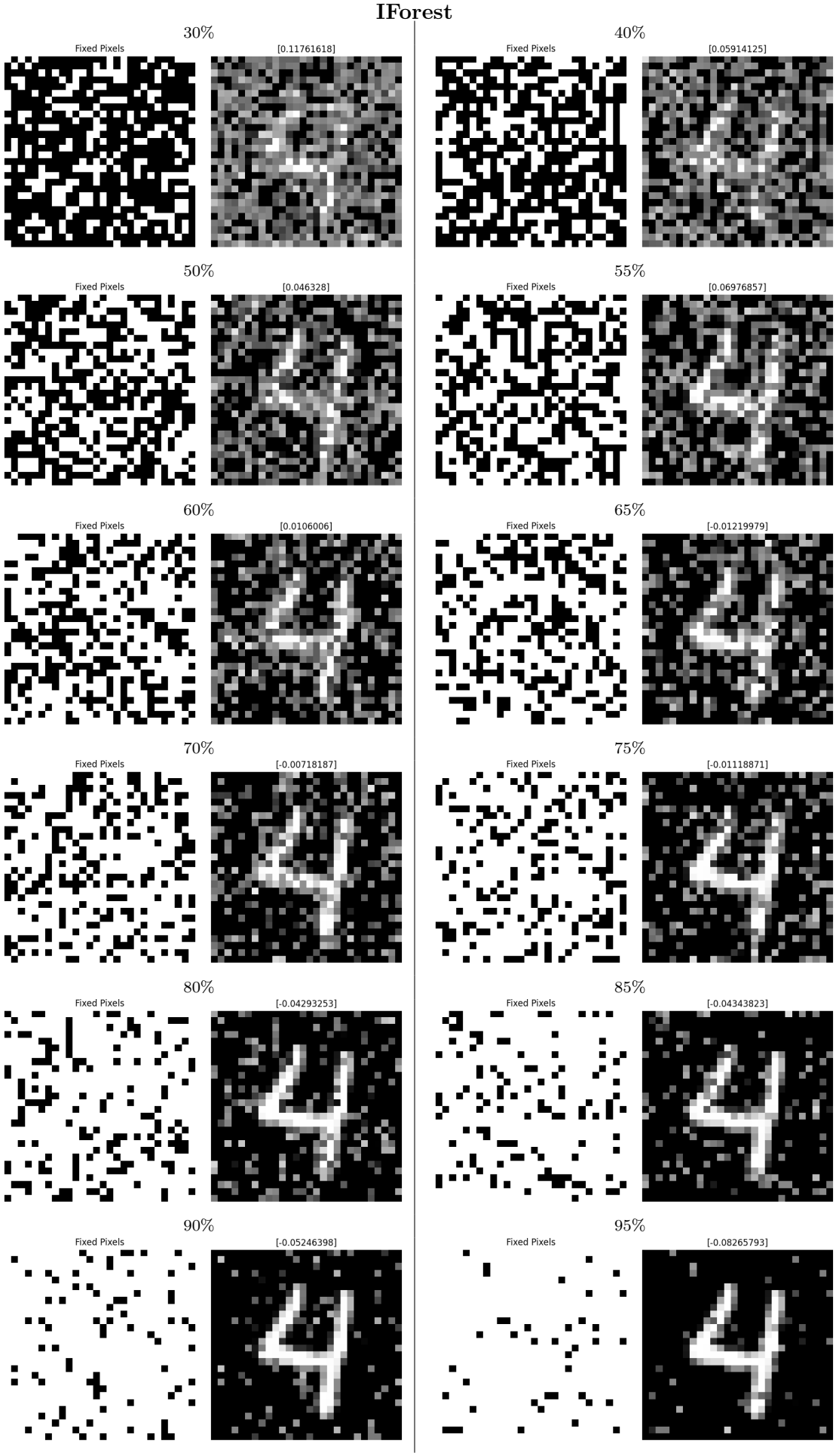

Figure A.6: Generated images on Isolation Forest

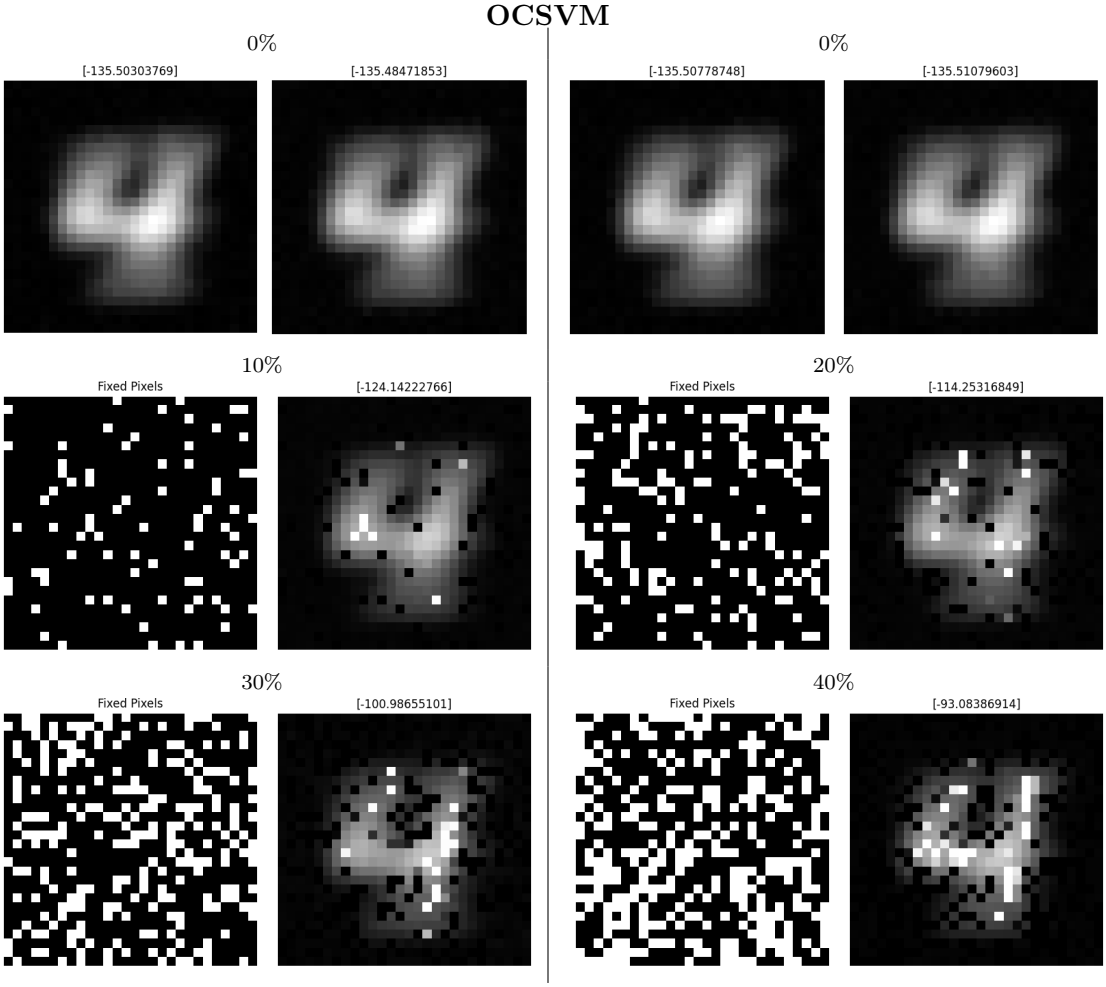

Figure A.7: Generated images on OCSVM. Top four 0% images are all generated images and have no fixed pixels. More images in Fig. A.8

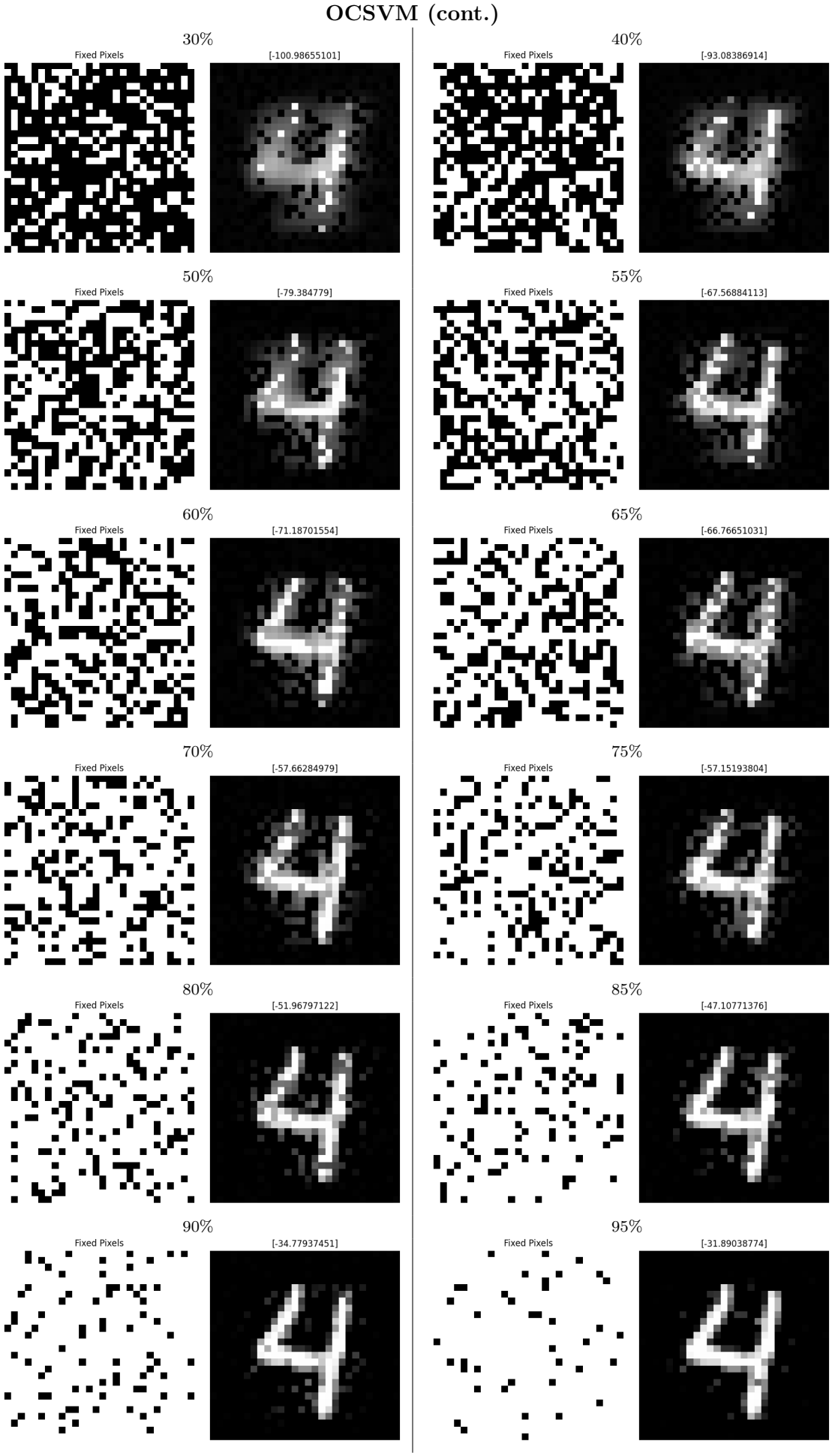

Figure A.8: Generated images on OCSVM (cont.)

List of Figures

Bibliography

[1] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. Software available from tensorflow.org.

[2] Aggarwal, Charu C.: *Outlier Analysis*. In *Data Mining*, chapter 2, pages 75–79. Springer, 2015.

[3] Aggarwal, Charu C and Charu C Aggarwal: *An introduction to outlier analysis*. Springer, 2017.

[4] Bengio, Yoshua, Aaron Courville and Pascal Vincent: *Representation Learning: A Review and New Perspectives*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798–1828, 2013.

[5] Bradley, Andrew P: *The use of the area under the ROC curve in the evaluation of machine learning algorithms*. Pattern recognition, 30(7):1145–1159, 1997.

[6] Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al.: *Language models are few-shot learners*. Advances in neural information processing systems, 33:1877–1901, 2020.

[7] Chandola, Varun, Arindam Banerjee and Vipin Kumar: *Anomaly detection: A survey*. ACM computing surveys (CSUR), 41(3):1–58, 2009.

[8] Deng, Li: *The mnist database of handwritten digit images for machine learning research [best of the web]*. IEEE signal processing magazine, 29(6):141–142, 2012.

[9] Developers, Scikit-Learn: *SVM Outlier Detection.* `http://scikit-learn.org/stable/modules/svm.html#svm-outlier-detection`, 2025. Accessed: 2025-03-13.

[10] Eiben, Agoston E and James E Smith: *Introduction to evolutionary computing.* Springer, 2015.

[11] Esteva, Andre, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau and Sebastian Thrun: *Dermatologist-level classification of skin cancer with deep neural networks.* nature, 542(7639):115–118, 2017.

[12] Gatys, Leon A, Alexander S Ecker and Matthias Bethge: *A neural algorithm of artistic style.* arXiv preprint arXiv:1508.06576, 2015.

[13] Goldstein, Markus and Andreas Dengel: *Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm.* KI-2012: poster and demo track, 1:59–63, 2012.

[14] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio: *Generative adversarial nets.* Advances in neural information processing systems, 27, 2014.

[15] Hanley, James A and Barbara J McNeil: *The meaning and use of the area under a receiver operating characteristic (ROC) curve.* Radiology, 143(1):29–36, 1982.

[16] Harshvardhan, GM, Mahendra Kumar Gourisaria, Manjusha Pandey and Siddharth Swarup Rautaray: *A comprehensive survey and analysis of generative models in machine learning.* Computer Science Review, 38:100285, 2020.

[17] He, Zengyou, Xiaofei Xu and Shengchun Deng: *Discovering cluster-based local outliers.* Pattern recognition letters, 24(9-10):1641–1650, 2003.

[18] Hinton, Geoffrey E and Ruslan R Salakhutdinov: *Reducing the dimensionality of data with neural networks.* science, 313(5786):504–507, 2006.

[19] Ho, Jonathan, Ajay Jain and Pieter Abbeel: *Denoising diffusion probabilistic models.* Advances in neural information processing systems, 33:6840–6851, 2020.

[20] Kingma, Diederik P, Max Welling et al.: *Auto-encoding variational bayes*, 2013.

[21] Klüttermann, Simon and Emmanuel Müller: *DEAN: Deep Ensemble Anomaly Detection.* `https://github.com/KDD-OpenSource/DEAN`, 2022.

[22] Krizhevsky, Alex, Geoffrey Hinton et al.: *Learning multiple layers of features from tiny images.* 2009.

[23] Lala, Sayeri, Maha Shady, Anastasiya Belyaeva and Molei Liu: *Evaluation of mode collapse in generative adversarial networks.* High performance extreme computing, 2018.

[24] Li, Zheng, Yue Zhao, Nicola Botta, Cezar Ionescu and Xiyang Hu: *COPOD: Copula-Based Outlier Detection.* In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1118–1123, 2020.

[25] Liu, Fei Tony, Kai Ming Ting and Zhi-Hua Zhou: *Isolation forest.* In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[26] O'shea, Keiron and Ryan Nash: *An introduction to convolutional neural networks.* arXiv preprint arXiv:1511.08458, 2015.

[27] Pang, Guansong, Chunhua Shen, Longbing Cao and Anton Van Den Hengel: *Deep learning for anomaly detection: A review.* ACM computing surveys (CSUR), 54(2):1–38, 2021.

[28] Radford, Alec, Luke Metz and Soumith Chintala: *Unsupervised representation learning with deep convolutional generative adversarial networks.* arXiv preprint arXiv:1511.06434, 2015.

[29] Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen and Ilya Sutskever: *Zero-shot text-to-image generation.* In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[30] Ruff, Lukas, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller and Marius Kloft: *Deep one-class classification.* In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[31] Sabuhi, Mikael, Ming Zhou, Cor-Paul Bezemer and Petr Musilek: *Applications of generative adversarial networks in anomaly detection: A systematic literature review.* Ieee Access, 9:161003–161029, 2021.

[32] Sakurada, Mayu and Takehisa Yairi: *Anomaly detection using autoencoders with nonlinear dimensionality reduction.* In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014.

[33] Schlegl, Thomas, Philipp Seeböck, Sebastian M Waldstein, Georg Langs and Ursula Schmidt-Erfurth: *f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks.* Medical image analysis, 54:30–44, 2019.

[34] Schölkopf, Bernhard, John C Platt, John Shawe-Taylor, Alex J Smola and Robert C Williamson: *Estimating the support of a high-dimensional distribution.* Neural computation, 13(7):1443–1471, 2001.

[35] Sekimoto, Kaiji and Muneki Yasuda: *Improving Interpretability of Scores in Anomaly Detection Based on Gaussian-Bernoulli Restricted Boltzmann Machine.* arXiv preprint arXiv:2403.12672, 2024.

[36] Shorten, Connor and Taghi M Khoshgoftaar: *A survey on image data augmentation for deep learning.* Journal of big data, 6(1):1–48, 2019.

[37] Tax, David MJ and Robert PW Duin: *Support vector data description.* Machine learning, 54:45–66, 2004.

[38] The MathWorks, Inc.: *What Is an Autoencoder?* `https://www.mathworks.com/discovery/autoencoder.html`. Accessed: 2025-03-12.

[39] Wu, Zhichao, Xin Yang, Xiaopeng Wei, Peijun Yuan, Yuanping Zhang and Jianming Bai: *A self-supervised anomaly detection algorithm with interpretability.* Expert Systems with Applications, 237:121539, 2024.

[40] Zhao, Yue, Zain Nasrullah and Zheng Li: *PyOD: A Python Toolbox for Scalable Outlier Detection.* Journal of Machine Learning Research, 20(96):1–7, 2019.

[41] Zhou, Chong and Randy C Paffenroth: *Anomaly detection with robust deep autoencoders.* In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

Eidesstattliche Versicherung

(Affidavit)

Ernst, Ben

Name, Vorname
(surname, first name)

232340

Matrikelnummer
(student ID number)

☒ Bachelorarbeit
(Bachelor's thesis)

☐ Masterarbeit
(Master's thesis)

Titel
(Title)

Evaluating Anomaly Detection Algorithms by Generating Samples from Anomaly Scores

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Ennepetal, 21.03.2025

Ort, Datum
(place, date)

B. Ernst

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Ennepetal, 21.03.2025

Ort, Datum
(place, date)

B. Ernst

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**