UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Unsupervised Anomaly detection

Author:
Valerii SAVOSKIN

Supervisor:
Oles DOBOSEVYCH

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, Valerii SAVOSKIN, declare that this thesis titled, "Unsupervised Anomaly detection" 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:

"Windmill or no windmill, he said, life woukd go on as it had always gone on- that is, badly"

George Orwell

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Unsupervised Anomaly detection

by Valerii SAVOSKIN

Abstract

The only way for the world to move into the bright future is to move from non-renewable resources into renewable ones. Creating and maintaining new economic spheres always requires human care and supervision, the magnitude of which can be lowered by using machine learning techniques. This work demonstrates the models that are created to solve the task of anomaly detection in an unsupervised fashion. This kind of methodology imposes a lot fewer restrictions on the data used while providing a framework to find cracks on a wind turbine. Moreover, it is a good building block for later research of unsupervised anomaly detection in the fields, where getting data might cost a lot, and the cost of mistakes is high. The success of the work can reduce the amount of time, money, and human resources for the big companies that utilize green energy and invest in the future of our planet.

Acknowledgements

First of all, I want to express my gratitude to company "COVIZMO" for given inspiration, ideas, support, computational resources and, the most important, datasets for this work.
Then I want to say thank you to my girlfriend who supported me along the way and put commas in the places they are supposed to be and mom who constantly reminded that my thesis must be finished.

Contents

List of Figures

List of Abbreviations

NN	**N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
AE	**A**autoEncouder
VAE	**V**ariational **A**autoEncouder
AAE	**A**dversarial **A**autoEncouder
CGAN	**C**onditional **G**enerative **A**dversarial **N**etwork
FCNN	**F**ully **C**onnected **N**eural **N**etwork

Dedicated to my mom and dad

Chapter 1

Introduction

1.1 Motivation

The invention of mechanisms for the use of renewable energy resources is a big step and a significant achievement of the 20th century. The tremendous 21st-century challenge is moving entirely from non-renewable resources into renewable ones and developing and improving the renewable energy industry. Renewable energy sources include solar radiation, wind, tides, waves, geothermal heat, etcetera. Generally, those are periodic or constant streams of energy, which are reproduced in nature and limited only by the stability of the Earth as a planet.
The benefits of using renewable resources are difficult to overestimate. These include a positive impact on the economy and reducing pollution. An excellent way for the development of humankind, in general, is to build a system of production and consumption that will make people independent of temporary and limited resources. Green energy is a crucial way to achieve this purpose.
One of the most popular renewable sources is the energy of wind, which specializes in the use of kinetic wind energy. Wind turbines convert the wind kinetic energy into mechanical power, which can be used for numerous tasks or may be converted into electricity by electric generators.
Operational experience demonstrated the advantages but also showed significant shortcomings of existing wind energy systems. The problem is that energy transformations in wind power systems occur with the help of massive moving elements (wind turbines); this causes their high need for periodic checkout and repairing throughout their service life. The primary motivation for writing this work is an attempt to solve the problem of expensive and time-consuming maintenance and repair of wind turbines through machine learning.

1.2 Wind turbines

Many different kinds of wind turbines are extensively used to generate electric power from the wind's kinetic energy. First of all, turbines might be offshore and onshore. They are a bit different in construction, but what is important to us is that it takes a lot more money and trained professionals to take care of the offshore turbines. In another way, turbines can be classified by the kind of blades that they use. Images that it has been worked with were of HAWT-type turbines.

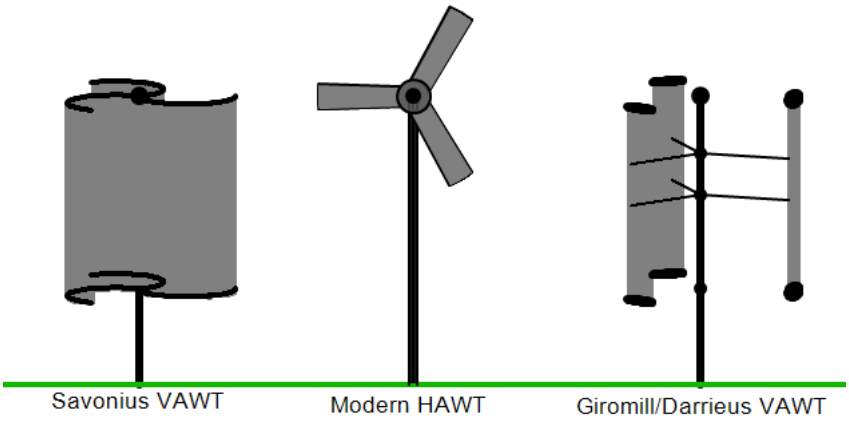


FIGURE 1.1: Types of turbines

These kinds of turbines consist of three main parts: blades, rotor, and tower. Because of severe wind, animals, water, and other reasons, blades and sometimes rotor might get damaged. Fixing the fault might save much money for the turbine owners compared to repairing the turbine when it gets damaged badly. Moreover, in the case of offshore turbines, the cost of repair dramatically depends on the cost of getting to the turbine. Hence, businesses are interested in decreasing the number of times the turbine gets inspected.

Chapter 2

Related works

In this chapter, we describe the works that investigated anomaly detection, and at the same time, we will look at the development of the GAN's.

2.1 Anomaly detection

Anomaly detection is a central area of interest within the field of machine learning and a classical problem in computer vision. Hence, anomaly detection and, more specifically - unsupervised anomaly detection is a question of great interest, and many studies were conducted about it.
Except for what has already been mentioned, there are such research directions in the public eye:
• financial: [Ahmed, Mahmood, and Islam, 2016],
• network systems [Ahmed, Mahmood, and Hu, 2016],
• fraud detection - [Abdallah, Maarof, and Zainal, 2016],
• biomedical [Schlegl et al., 2017],
• security such as video surveillance [Kiran, Thomas, and Parakkal, 2018].

Methods of solving anomaly detection could be classified into five main categories [Basora, Olive, and Dubot, 2019]:
1. Reconstruction based methods;
2. Domain-based methods;
3. Statistical methods;
4. Ensemble-based methods;
5. Distance-based methods.

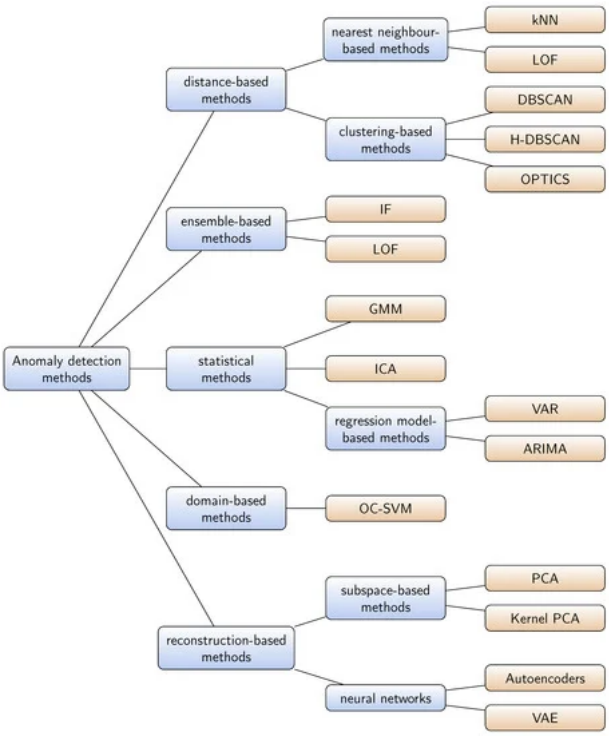


FIGURE 2.1: Anomaly detection methods

Distance-based methods include nearest neighbours-based methods (kNN, LOF) [Kriegel et al., 2009] and clustering-based (DBSCAN, HDBSCAN) [Budalakoti, Srivastava, and Otey, 2009]. All methods of this category rely on some kind of distance between points, thus treating every instance as a point, which might not be completely true, as it will not work that well with complex data, for example, images, time series, etcetera.

When speaking of ensemble methods, the first thing that comes to our minds - *random forest*. Anomaly detection has a modification of it called - *Isolation Forest* [Liu, Ting, and Zhou, 2008]. Shortly, it uses the distance between the root of the forest to the sample location. This method is very computationally efficient and sometimes might even challenge *the NN's*.

Statistical methods mainly rely on estimates of the probability density function and later assume that *normal* data will have a decent probability and *abnormal* will have small probability chances. Gaussian Mixed Models [Pontoppidan and Larsen, 2003] work well to estimate the probability density function, though it has a pitfall trying to estimate the function using all data, thus also trying to fit into anomalies. So if data consists of many anomalies, the function might also be shifted to include them. Independent component analysis is also used as a statistical method. It assumes that data are mutually exclusive, non-normally distributed samples, and samples are independent. The last subcategory of statistical methods is regression models. They are highly used in time series analysis. The detection consists of two steps - first, fitting the model into the data and then testing it on a test sequence to find residuals between the model's prediction and real value. Autoregressive Integrated Moving Average and Vector Auto-Regressive are regression models.
Domain-based methods use the assumption that *normal* data is "normal" in the sense that it is possible to create a boundary between *normal* and *abnormal* data using linear or non-linear methods. Support Vector Machine [Schölkopf et al., 1999]

is used the most frequently, but simple models like Logistic Regression might also be used.
Furthermore, reconstruction-based approaches - their main idea is that squishing data into lower-dimensional representation might help to differentiate between *normal* and *abnormal* samples. That is, some kind of information might be lost while deconstructing *abnormal* data. It can be classified into two parts - subspace-based methods and *NN* methods.
Subspace-based are based on PCA(principal component analysis) [Jolliffe and Cadima, 2016]. Those models benefit from being relatively small and easy to use, but PCA itself provides some pitfalls, like linearity and noise sensitivity, that could be changed using PCA modifications - Kernel PCA, Robust PCA, etcetera. *NN* models are using *NN's* ability to squeeze data while providing non-linearity. Those methods include *AE, VAE*, all kinds of *GANs*. Models that were used in this thesis are reconstruction-based *NN* models.

2.2 Reconstruction based anomaly detection

Because the subspace-based approach has its limitations, reconstruction-based anomaly detection is now the most researched area of machine learning. This approach would not be feasible without the development of whole generative modeling that started from *AE* [Ballard, 1987], went through *VAE* [Kingma and Welling, 2014] and later on developed into *GAN* [Goodfellow et al., 2014]. During that evolution, there were attempts to turn the generative model into *an anomaly detection generative* model. Researchers tried their best to use simple *AEs* [Sakurada and Yairi, 2014] to use the non-linear nature of *NN's* and compare generated image $\hat{X}$ and original images X. Another case of interesting *AE* usage - [Gong et al., 2019], here attempt was made to create a memory that would modify encoded z vector, depending on previously learned features. Later on, *VAE* took its turn on *anomaly detection* task [Lupo, 2019], and their modifications that introduced stochasticity [Pol et al., 2020] and a mixture of *VAE* and *convolutional autoencoders* [Yu, Kavitha, and Kurita, 2020]. Nevertheless, the most promising results were received using *GAN* modifications. Many pieces of research were based on creating an anomaly score based on the difference between an original image and a restored image.[Akçay, Atapour-Abarghouei, and Breckon, 2019], [Zenati et al., 2018]. A current state-of-the-art approach that only recently got beaten - *EGBAD* [Zenati et al., 2018] utilized the *BiGAN* architecture, which allowed training GAN that makes *decoder* invert *the encoder*, which made a great impact because of its mathematical basis [Donahue, Krähenbühl, and Darrell, 2016]. Another great impact paper - *AnoGAN*[Schlegl et al., 2017], introduces a multistep learning algorithm and a complex anomaly score.
Interesting attempts were made by *Ravanbakhsh* et al.[Ravanbakhsh et al., 2017] who used two *CGAN's*, the first one of which produced optical frames and the second one regenerated images from optical frames.

As for now, *generative models* seem to show significant results in *the anomaly detection* area because of their high score and *unsupervised* nature of learning. While still suffering from *generative model* problems - they might be hard to train because of numerical instability, and the metric by which the *abnormality* of the image must be conducted is unobvious and must be carefully designed by each individual researcher.

Chapter 3

Background information

3.1 NN

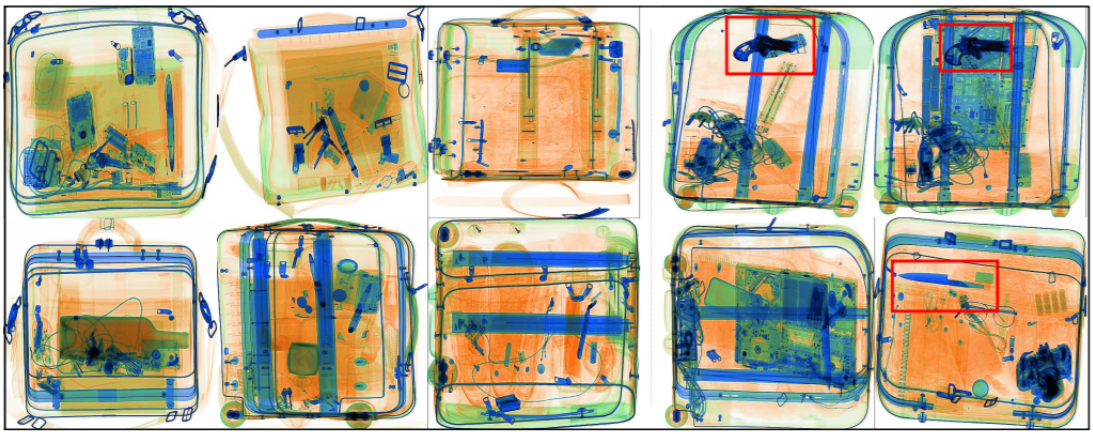
Further models can not be introduced without an understanding of what the neural network is. This concept can be viewed from different perspectives, and many intuitions could be applied to it, from neurons of our brain to a huge nonlinear function. To put it simply, *NNs* are a set of connected units used to find hidden dependencies and insights into the data. That is, it "learn" something from the data, depending on the task given.
This chapter explains *the NN* on the example of *multilayer perceptron* or, as it is called now - *FCNN*, a model that kickstarted the whole machine learning industry back in the 1960s. The perceptron consists of building blocks that are called layers, which in turn consist of neurons. This idea was inspired by the work of human brains, though in practice, it is a lot different from it. Each neuron has connections to a previous and the following layer, using which the weights are "transported." Also, nowadays, neurons have activation functions built-in for the sake of non-linearity. Without them, *the NN* would turn into a simple matrix multiplication. The signals start at the input stage; then it traverses forward up to the last layer. To later train *the perceptron*, the derivative of each weight is taken using the chain rule, and the weights are updated using those gradients.
In modern days machine learning researchers invented more kinds of layers. These are *Convolutional layers*, *Recurrent layers*, *Batch Normalization layers*, *Dropout layers*, and lots and lots more. This chapter will briefly explain only the layers used in the models that the author put his hands on.
1. Convolutional layers - 1, 2, 3, or even more dimensional layers are used to process high dimensional data, such as images, videos, etcetera. They were applied in a filtered manner, after filtering might come to activation as in *FCNN*.
2. Transpose convolution - reversed operation of convolution, used for upsampling from vector to an original image. 3. Batch normalization - during the training phase, values of activations might be very different in magnitude, which makes it harder for the model to learn. Batch normalization helps to fix this problem by learning *the mean* and *standard deviation* of the given data and normalizing it using them.

3.2 Anomaly detection

Anomaly detection is a machine learning task in which the main goal is set to identify data points that are in some way different from other data points. Meanwhile, it is one of the most classic and applied computer vision tasks, which is exciting and

still opened. Its main distinction from other computer vision tasks - high-class disbalance. This means that we usually have many samples of *normal* class and a lot fewer samples of *abnormal* class.

To provide some concrete examples, let us use images of "University Baggage Anomaly Dataset," which is currently private because of security reasons, but it still works well as a demonstration. Taken from [Samet Akcay, 2018]



(a) Normal Data (X-ray Scans) (b) Normal + Abnormal Data (X-ray Scans)

FIGURE 3.1: Images of "University Baggage Anomaly Dataset"

As it can be seen, *normality*, in this case, means the absence of a weapon or any other dangerous item in the suitcase.

3.3 Generative models

Another concept that one might grasp to understand the following models is a generative model and its difference from a discriminative model.
Discriminative models learn P(Y | X), namely given some parameters of the probability of X belonging to class Y, where X - is our data and Y - classes. On the other hand, generative models learn P(X, Y), that is the joint probability of X and Y. That is why having a joint probability, we can generate an instance of X while knowing what the likelihood of them being Y is. It should also be stated that generally, it is harder to create a generative model than to create a discriminative model.

In this thesis, it has been used a special kind of generative model - *GAN*. However, to explain it, it is better to start with its logical ancestor - *AE* and *VAE* to make the main idea easier to grasp.

3.4 AE

AE - is an Encoder-Decoder type of model. It consists of two parts - first, that turns the initial high dimensional image into a squeezed vector representing the most distinctive features of a given data.

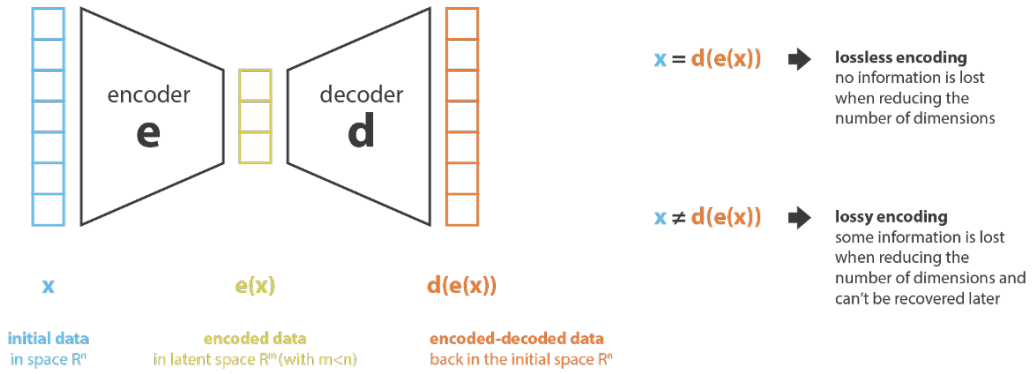


FIGURE 3.2: Autoencoder's architecture

Its first purpose is to create a model that can learn a lower-dimensional representation of the data, that is, to create a dimensionality reduction pipeline that uses nonlinear features.
The central intuition behind choosing this kind of model is that they can easily be trained to restore an image from a vector using the *L2* metric to compare x and $d(e(x))$. However, that way, the model can set all the weights to some kind of identity matrix. A more challenging task arises later when we understand that the model should be able to generalize in some way, that is, to learn how to restore concrete images that it was trained on, but restore even the one that it has not yet seen.
For that sake, a couple of methodologies are used:
1. Carefully choosing our network layers and neurons to prevent them from overfitting.
2. Construct out loss function, such that it penalizes the activation of neurons. That way, we encourage the network to use as few neurons as possible. It can be done either by adding *L1* of all activation or *KL-divergence* between each neuron and *Bernoulli RV* that models "perfect" neuron activation to a loss function.
3. Adding noise to an input image, our model cannot directly "remember" the images because the output image is different from an input image.

3.5 VAE

The main disadvantage of *AE*, which later led to the creation of *VAE*, is the inability to create new data. We might be tempted to think that sampling some point from the distribution of the z vector might create something new. However, in practice, it rarely works because of model architecture, initial data distribution, etcetera, which makes it almost impossible to predict whether this method will actually work. So what is the difference between *VAE* and *AE* - *VAEs* were built to ensure that models do not overfit, and its latent space has properties that would later enable the new data generation process. That is, instead of learning representation $z = e(x)$, it learns $p(z \mid x)$ - the conditional distribution of latent space. The AE can be viewed as a deterministic version of *VAE*.
The training pipelines turns into a bit more sophisticated algorithm:
1. The input image is encoded as a distribution over the latent space
2. A point z is a sample from that distribution
3. This point goes through a decoder, and loss is calculated

4. Backpropagation with calculated loss occurs

Distributions are usually chosen to be gaussian to make them easier to work with. Except for the stochasticity update compared to *AE*, *VAE* also introduces a new kind of regularization. To prevent overfitting and behavior like AE, we penalize *mean* and *standard deviations* by adding *KL divergence* of it and *standard normal distribution* to the loss function. The last detail that should be added, because sampling is not differentiable, models tend to predict *the mean* and *standard deviation* of distributions in practice.

3.6 GAN

GAN is the last logical ancestor of *AE* that would be noted here. It takes a generation task to the next level by introducing two-stage learning and a game theory like loss function.
GAN consist of two models - *Generator* and *Discriminator*. Intuitively they can be treated as two players trying to fool each other; they even have the same loss function that one tries to minimize while the other tries to maximize.
The generator model consists of a *decoder* that takes a random vector and produces an entity, mostly an image. Later on, a *Discriminator* that consists of an *encoder* compares this value to an actual image that fits into it. All pipelines can be visualized like this:

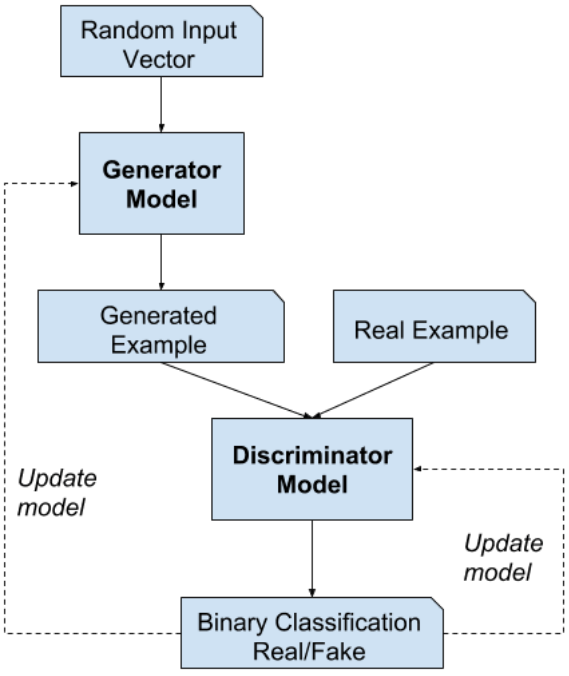


FIGURE 3.3: The general architecture of the Generative Adversarial Network

3.7 Problem formulation

More formally, our problem might set like that, given the dataset D, which contains numerous *normal* data X, and a smaller amount of *abnormal* data $\hat{X}$, create such

a model f, that has parameters θ, that would minimize the given loss function to further distinguish between X and $\hat{X}$. Because we used generative modeling, our model f should learn the proper distribution of *normal* samples, therefore, producing some kind of abnormality score $A(x)$, that after applying the threshold T will show whether an image x is *normal* or *abnormal*.
While adding a bit more Encoder-Decoder notation, it is necessary to mention that part of f that is responsible for mapping from R^n to R^z, or from x to z, will be named $f_e(x)$, and the function that maps z to $\hat{x}$ will be named $f_d(z)$.

Chapter 4

Proposed approach

In this thesis, several methods are tried to solve *unsupervised anomaly detection*, which will be described below shortly. Most of them are going to be based on GAN architecture.

4.1 Ganomaly

Ganomaly is an approach to an anomaly detection problem introduced by Samet Akcay et al [Samet Akcay, 2018]. It is a novel approach found in almost all anomaly detection surveys because of several new features that it introduced. As all *GANs*, Ganomaly consists of *Generator* and *Discriminator*. An illustration is taken from the paper:

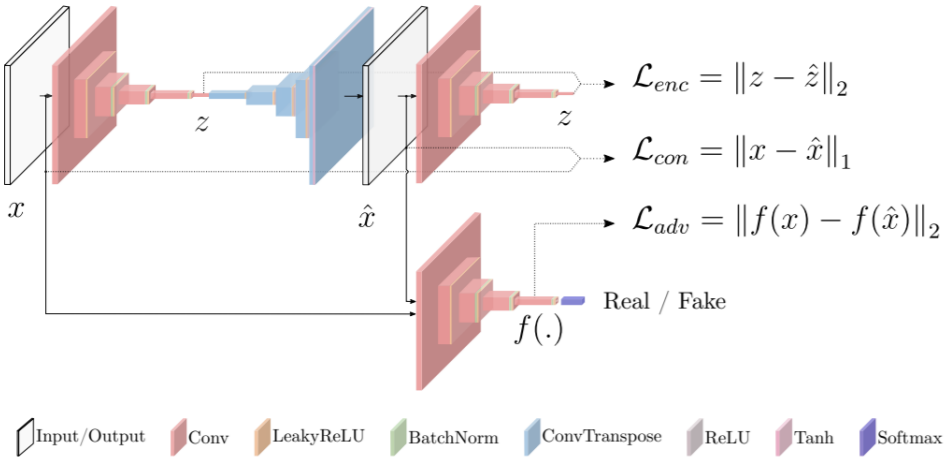


FIGURE 4.1: Ganomaly architecture

Generator(G) consists of *AAE* is responsible for encoding input image x into latent vector *z*, later decoding it into $\hat{x}$, and again encoding $\hat{x}$ into $\hat{z}$ for later usage. It consists of three subnetworks, two encoders, one decoder and can be seen on the upper part of the image.
Discriminator(D) consists of a simple *encoder* that discriminates *x* from $\hat{x}$ and that way training *generator*. Conceptually it is the same as in simple GAN. Their both building blocks are - *2D convolution* layers that *LReLU* follows in the case of the encoder and *ReLU* in the case of the decoder. *LReLU* is used to prevent gradient vanishing. Later Batch normalization layer is used.

The motivation of using many such subnetworks becomes apparent when we take a closed look at the loss functions that are used:

1. Adversarial loss - new research [Schlegl et al., 2017], [Zenati et al., 2018] shows that modifying generator training from using just binary D output to using D's internal representation helps with numerical stability while training the whole network. That is why instead of binary cross-entropy being used on the last sigmoid layer, it is used on the final convolutional layer's value when fake and true data are used as input. More strictly, it is written so:

$$\mathcal{L}_{adv} = \mathbb{E}_{x \sim p_{\mathbf{x}}} \| f(x) - \mathbb{E}_{x \sim p_{\mathbf{x}}} f(G(x)) \|_2.$$

2. Contextual loss - idea that was adopted from *AE*. Generated images must be similar to input images; that is why some kind of loss that compares them must be added. Empirically it was shown that *L1* works better than *L2* [Isola et al., 2017]; that is, it produces less blurry results. Mathematically:

$$\mathcal{L}_{con} = \mathbb{E}_{x \sim p_{\mathbf{x}}} \| x - G(x) \|_1.$$

3. Encoder loss, one of the new features that Ganomaly brought to the public, is the idea that *anomaly* images are restored worse than *normal* ones. This is what is used in encoder loss. This loss compares z and $\hat{z}$ using *the L2* metric. If a picture has *an anomaly* generator will not be able to restore it perfectly, thus making Encoder loss bigger.

$$\mathcal{L}_{enc} = \mathbb{E}_{x \sim p_{\mathbf{x}}} \| G_E(x) - E(G(x)) \|_2.$$

In train time, all of these losses are weighted and summed:

$$\mathcal{L} = w_{adv}\mathcal{L}_{adv} + w_{con}\mathcal{L}_{con} + w_{enc}\mathcal{L}_{enc}$$

Empirically the most efficient values of coefficients are:
1. $w_{adv} = 1$
2. $w_{con} = 50$
3. $w_{enc} = 1$

During training, the network is fed with only *normal* images to grasp the *normality* of the data. Because later on, we need to use the model to differentiate *normal* images from *abnormal* ones. Ganomaly was created to have an output that could be used as an abnormality score.
It is L_{enc}. Nevertheless, because the loss is unbounded bottom and top, we have to additionally normalize it to make sense of the values that it provides. It is done by subtracting minimum and dividing by range, that is:

$$s'_i = \frac{s_i - min(\mathcal{S})}{max(\mathcal{S}) - min(\mathcal{S})}$$

4.2 Skip-Ganomaly

Skip-Ganomaly is a modification of Ganomaly that came out a year later. It reworked Ganomalie's architecture by making it more straightforward but capable of learning more because of RES-NET like skip connections. A good image of the architecture:

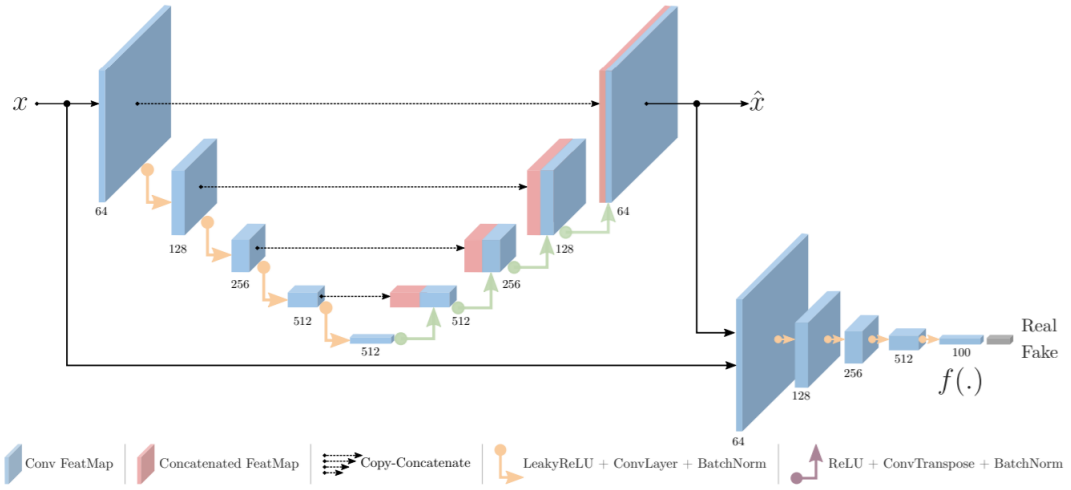


FIGURE 4.2: Architecture Skip-Ganomaly

The *generator* this time is more straightforward - it only has *AAE* in it. However, now every layer in the encoder has a connection to a corresponding layer in the decoder. That is the result of the n-th layer of the encoder gets concatenated to the output of the n-th layer of *the decoder*. Those skip connections allow gradients to flow more accessible through the model, which in turn allows building bigger models and making it easier for the model to learn.
Losses also got modified, now there are three losses:

1. Adverbial loss - now it is classical min-max binary cross-entropy loss. The *generator* gets updated depending on the output of the discriminator.

$$\mathcal{L}_{adv} = \mathbb{E}_{x \sim p_x}[logD(x)] + \mathbb{E}_{x \sim p_x}[log(1 - D\hat{x})].$$

2. Contextual loss - the same as in Ganomaly. Were built to enforce *generator* to create realistic images.

$$\mathcal{L}_{con} = \mathbb{E}_{x \sim p_x}|x - \hat{x}|_1.$$

3. Latent loss is an adverbial loss in Ganomaly. Utilizes the last Convolutional layer of *the decoder*.

$$\mathcal{L}_{lat} = \mathbb{E}_{x \sim p_x}|f(x) - f(\hat{x})|_2.$$

The removal of one of the *encoders* seems to show that it was actually redundant. The training procedure is the same as in Ganomaly, but testing changes a little due to the encoders' absence. Now it becomes:

$$\mathcal{A}(\dot{x}) = \lambda R(\dot{x}) + (1 - \lambda)L(\dot{x})$$

Where, $R(\dot{x})$ - contextual loss and $L(\dot{x})$ - latent loss. Once again, the score must be normalized:

$$\hat{\mathcal{A}}(\dot{x}) = \frac{\mathcal{A}(\dot{x}) - min(\mathbf{A})}{max(\mathbf{A}) - min(\mathbf{A})}.$$

Chapter 5

Experiment setup

5.1 Datasets description

In order to thoroughly test the abilities of our models, three datasets are used, ranging from the simplest MNIST [LeCun, 1999] to a bit harder CIFAR [Krizhevsky, Hinton, et al., 2009] and the real-world dataset.

5.1.1 MNIST

The MNIST database was constructed from NIST's Special Database 3 and Special Database 1, containing binary images of handwritten digits [LeCun, 1999]. The training part consists of 60,000 images and the test part of 10,000. Both sets are from approximately 250 writers, who are disjoint on train and test parts. Images are of size 28 by 28, and they are grayscale. This dataset is used for the classification task. Current state-of-the-art methods of solving these classification tasks can be found in [LeCun, 1999]. In the conducted experiment, one of the classes of MNIST was treated as *abnormal*, while others as normal. That is how, in total, ten sets of data were received, each containing a single digit as *an anomaly*.

5.1.2 CIFAR10

The CIFAR10 consists of 60,000 RGB images that are split into train and test parts. The training part consists of 50,000 pictures and testing of 10,000 pictures. CIFAR10 contains images of 10 different classes, namely: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images for each class, and classes are mutually exclusive. Here our usage of CIFAR comes to again picking one class as *abnormal* and others as *normal*.

5.1.3 Wind turbines dataset

The dataset on which the models were tested belongs to a company that works with wind turbines damages. It consists of images of wind turbine blades and sometimes rotors. Example of images:

FIGURE 5.1: Example of wind turbines dataset

All images were taken either by drone, by the camera on the ground, or by the camera attached to blades. Different companies supplied photos, so they might look different because they were shot on different cameras and in different places.

Abnormalities that should be found consist of different categories. Nevertheless, one of the problems of the wind turbines industry is that it does not have a unique categorizing standard. Every business develops its standard that highly depends on the types of wind turbines they use, their locations, etcetera. This research will try to stick to one of those standards. *Abnormalities* could be categorized by severity, time of occurrence, and location on the blade. Severity is the most critical metric because it shows whether a supplier needs to take care of the turbine, repair it sometimes, or even immediately stop the turbine. Each *abnormality* that is found is assigned a score(usually from 1 to 5, or from 1 to 10) that means how severe the damage is. The lower score, the less severe the damage is. Lower scores might represent paint damage or some kind of dirt that stuck to the turbine. Respectively high score means serious damage that might lead to turbine failure. It is important to find those damages, because some of them require an urgent treat.

Initially, the dataset consists of images shown above, but it is hard to train a model with a background taking a lot of picture space. So either a trained, supervised model that finds blades on images or pixelwise mappings that were provided either by the client or made by annotators, used to later color the background into black color. Example of black images:

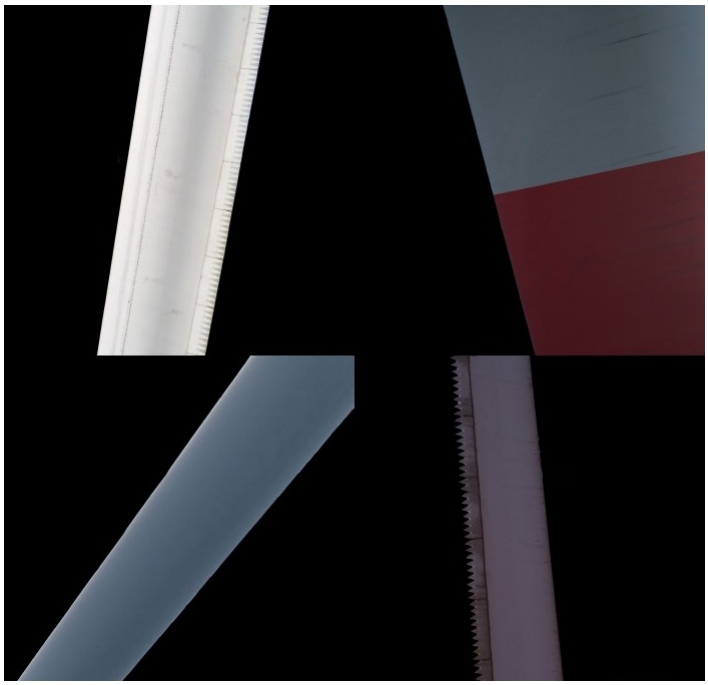


FIGURE 5.2: Wind turbines with removed background

After that images that are of size roughly 8000 by 6000 are cropped into 256 by 256 bits. They later the one that has defects on them are marked as *abnormal* and others are marked as *normal*.

5.2 Results

5.2.1 Mnist results

The metric that was measured for current and following - Area under curve(AUC). The picture below demonstrates the score for each model tested. One might notice that some of the classes have a lot lower AUC scores than others, it happens because of the nature of the data that is, some digits are harder to differentiate than others. All the models needed only a few epochs to train(1-3) due to their size and generalizing ability.

FIGURE 5.3: AUC of Mnist

It is also interesting to look at the distribution of *normal* and *abnormal* classes. It can be visually seen that means of the two distributions vary and instances of *abnormal* class have higher abnormality scores than *normal*.

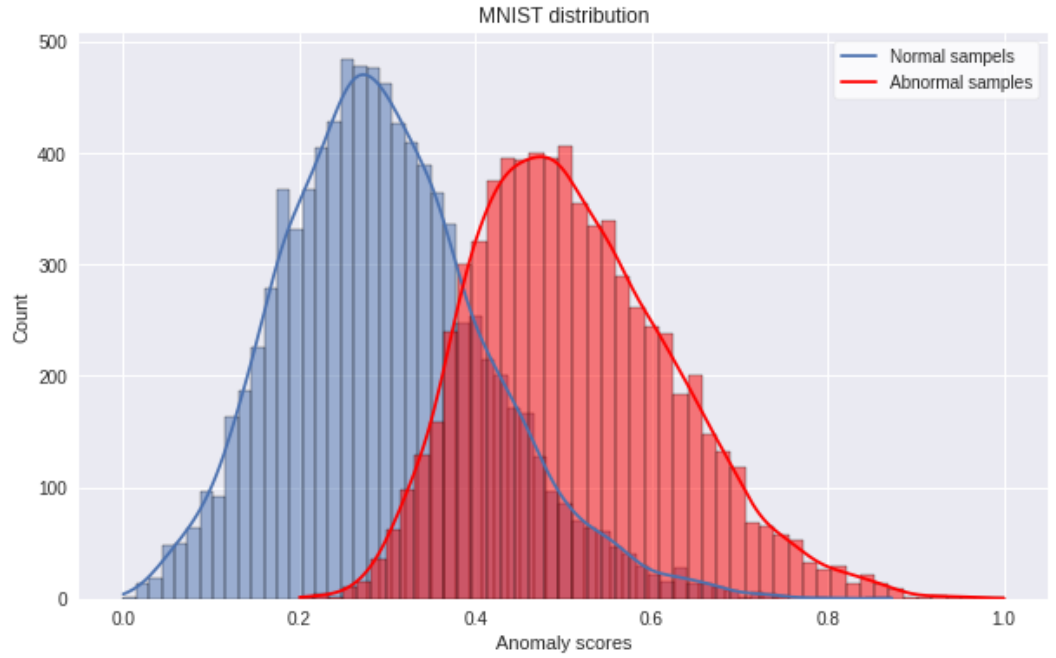


FIGURE 5.4: Abnormality scores distribution

5.2.2 CIFAR results

The score was again chosen to be an AUC. This time models perform worse than in Mnist because of CIFAR data complexity. Also, CIFAR unlike MNIST has all three channels - red, green and blue. Again like in the MNIST case, CIFAR has classes that

are harder to classify because of their complexity. All of the models were trained for a few epochs(1-3) with a low learning rate of 1e-4.

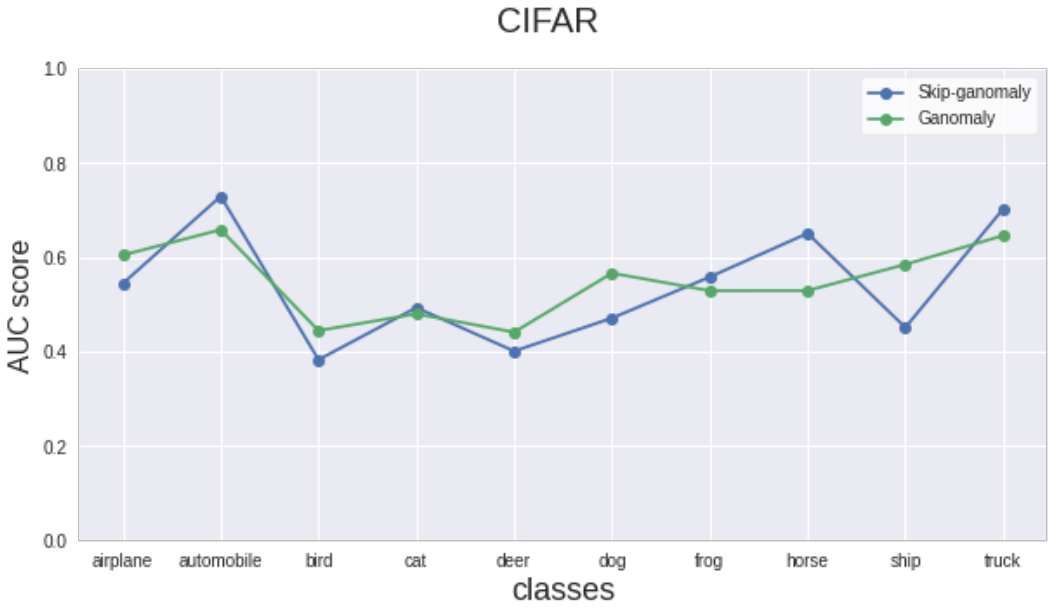


FIGURE 5.5: AUC of CIFAR

The distribution of classes now looks a bit different. The average AUC score on CIFAR is worse than on MNIST so we might conclude that models learned to differentiate classes worse. This is the consequence of a more smoothed distribution.

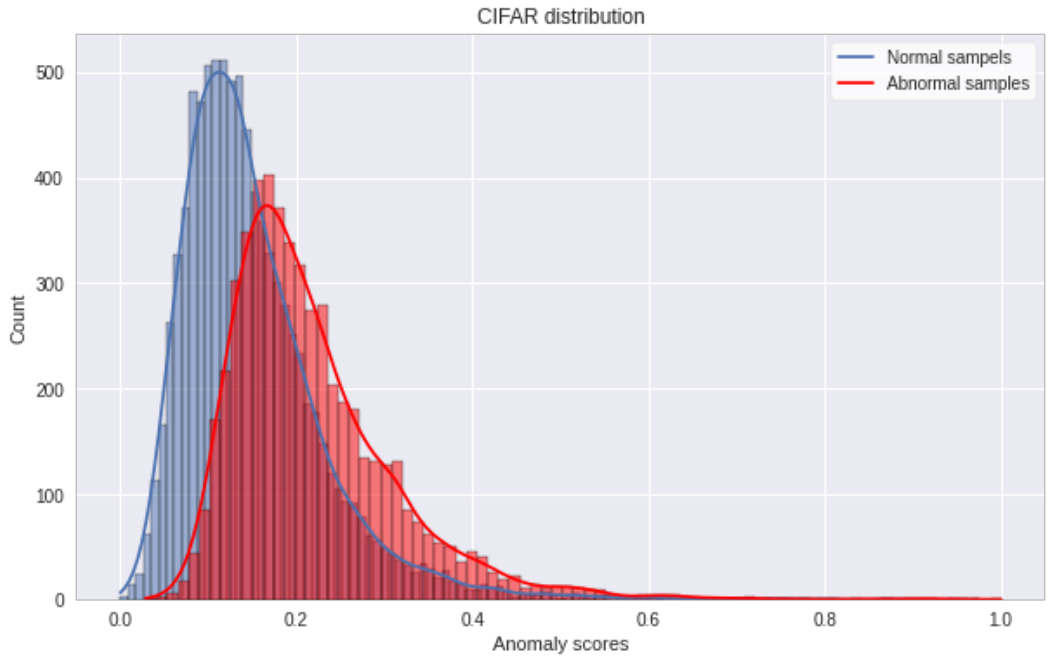


FIGURE 5.6: Abnormality scores distribution

5.2.3 Wind turbines results

Both models were trained on 4k training dataset and tested on 750 no defects images and 500 defects images.
The distribution of scores on Ganomaly looks so:

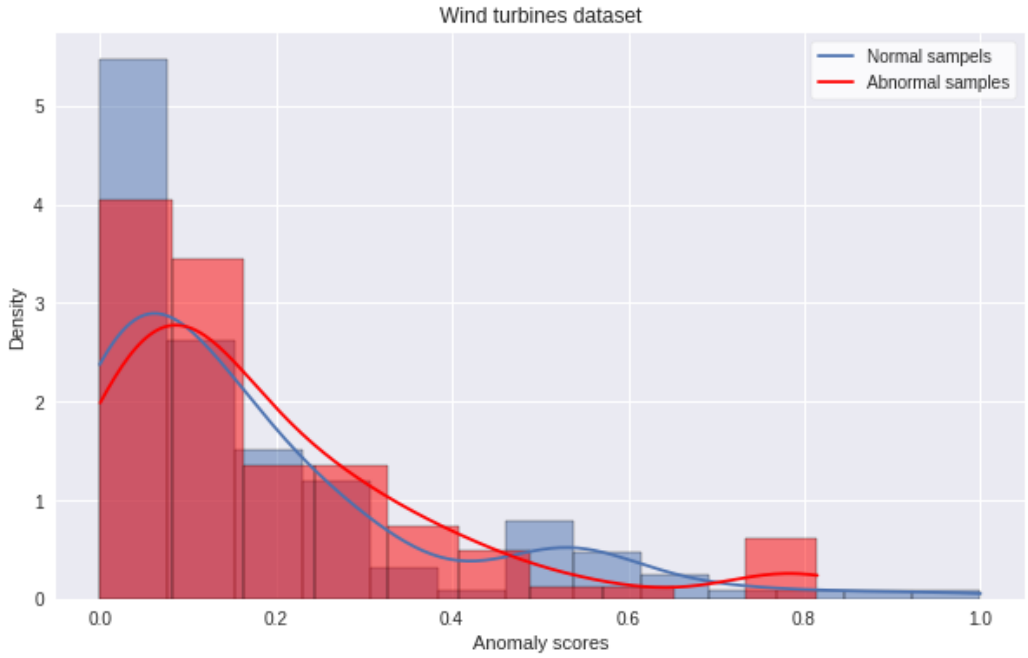


FIGURE 5.7: Ganomaly abnormal score distribution

And Skip-Ganomaly:

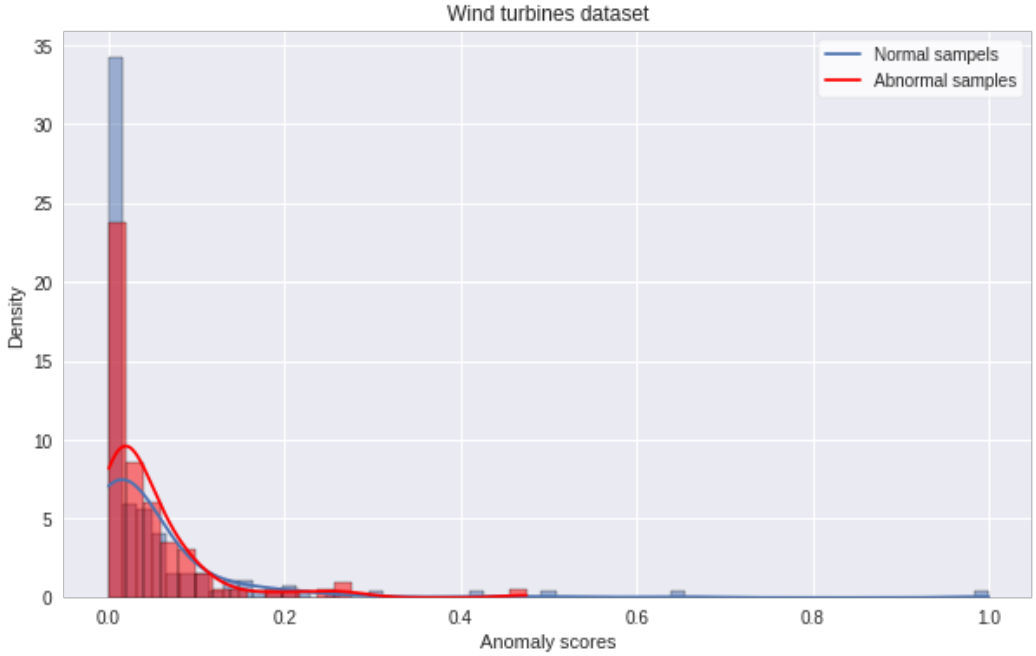


FIGURE 5.8: Skip-Ganomaly abnormal score distribution

Though the distribution on the Ganomaly looks better, it shows worse results:

- Ganomaly - 0.57
- Skip-Ganomaly - 0.62

Chapter 6

Conclusion and future works

In this paper, the unsupervised methods of solving anomaly detection tasks were presented. All of them were based on GAN architecture.
Unsupervised learning is a promising approach to that task and despite yet a gap between its performance and supervised learning performance. This kind of learning is going to be later researched because there are tasks for which supervised learning is impossible.
We managed to train GAN on different datasets, despite numerical instability by using different kinds of the loss function. From the author's experience, GAN's are going to move from the game theory like loss function to ones that are task or architecture-specific. Because classical GAN [Goodfellow et al., 2014] and its modification that uses the game theory lie loss function are a lot harder to train than the one that is presented in this paper. The author's implementation of the models used and described in this paper is available here.

The future work will include trying different, probably bigger models and investigating more into unsupervised learning anomaly detection, especially of generative type. Also, in practice, the statement that decoder decodes abnormal data worse than normal does not always hold true, so the job of implementing Encoder-Decoder, which would have a "memory," has already started.

Bibliography

Abdallah, Aisha, Mohd Aizaini Maarof, and Anazida Zainal (2016). "Fraud detection system: A survey". In: *Journal of Network and Computer Applications* 68, pp. 90–113. DOI: 10.1016/j.jnca.2016.04.007.

Ahmed, Mohiuddin, Abdun Naser Mahmood, and Jiankun Hu (2016). "A survey of network anomaly detection techniques". In: *Journal of Network and Computer Applications* 60, pp. 19–31. DOI: 10.1016/j.jnca.2015.11.016.

Ahmed, Mohiuddin, Abdun Naser Mahmood, and Md. Rafiqul Islam (2016). "A survey of anomaly detection techniques in financial domain". In: *Future Generation Computer Systems* 55, pp. 278–288. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2015.01.001. URL: https://www.sciencedirect.com/science/article/pii/S0167739X15000023.

Akçay, Samet, Amir Atapour-Abarghouei, and Toby P. Breckon (Jan. 25, 2019). "Skip-GANomaly: Skip Connected and Adversarially Trained Encoder-Decoder Anomaly Detection". In: arXiv: 1901.08954 [cs.CV].

Ballard, Dana H. (1987). "Modular learning in neural networks". In: *AAAI, Department of Computer ScienceUniversity of Rochester, Rochester, New York*. URL: https://www.aaai.org/Papers/AAAI/1987/AAAI87-050.pdf.

Basora, Luis, Xavier Olive, and Thomas Dubot (2019). "Recent Advances in Anomaly Detection Methods Applied to Aviation". In: *Aerospace* 6.11, p. 117. DOI: 10.3390/aerospace6110117. URL: https://doi.org/10.3390/aerospace6110117.

Budalakoti, S., A.N. Srivastava, and M.E. Otey (2009). "Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39.1, pp. 101–113. DOI: 10.1109/tsmcc.2008.2007248. URL: https://ieeexplore.ieee.org/abstract/document/4694106.

Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell (May 31, 2016). "Adversarial Feature Learning". In: arXiv: 1605.09782 [cs.LG].

Gong, Dong et al. (Apr. 4, 2019). "Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection". In: arXiv: 1904.02639v2 [cs.CV].

Goodfellow, Ian J. et al. (June 10, 2014). "Generative Adversarial Networks". In: arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/pdf/1406.2661.pdf.

Isola, Phillip et al. (2017). "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2017.632. URL: https://ieeexplore.ieee.org/document/8100115.

Jolliffe, Ian T. and Jorge Cadima (2016). "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065, p. 20150202. DOI: 10.1098/rsta.2015.0202. URL: https://royalsocietypublishing.org/doi/full/10.1098/rsta.2015.0202.

Kingma, Diederik P and Max Welling (2014). "Auto-Encoding Variational Bayes". In: arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/pdf/1312.6114.pdf.

Kiran, B Ravi, Dilip Mathew Thomas, and Ranjith Parakkal (Jan. 9, 2018). "An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos". In: arXiv: 1801.03149 [cs.CV]. URL: https://arxiv.org/abs/1801.03149.

Kriegel, Hans-Peter et al. (2009). "LoOP". In: *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*. ACM Press. DOI: 10.1145/1645953.1646195. URL: https://dl.acm.org/doi/abs/10.1145/1645953.1646195.

Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). "Learning multiple layers of features from tiny images". In: *Tech. rep., Citeseer*. URL: http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

LeCun Yann, Cortes Corinna Christopher J.C. Burges (1999). *The MNIST database of handwritten digits*. URL: http://yann.lecun.com/exdb/mnist/.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008). "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. DOI: 10.1109/icdm.2008.17. URL: https://ieeexplore.ieee.org/abstract/document/4781136.

Lupo, Francesco (2019). "Variational Autoencoder forunsupervised anomaly detection". MA thesis. Politecnico di Torino, Faculty of Engineering. URL: https://webthesis.biblio.polito.it/10360/1/tesi.pdf.

Pol, Adrian Alan et al. (Oct. 12, 2020). "Anomaly Detection With Conditional Variational Autoencoders". In: arXiv: 2010.05531 [cs.LG]. URL: https://arxiv.org/pdf/2010.05531.pdf.

Pontoppidan, N.H. and J. Larsen (2003). "Unsupervised condition change detection in large diesel engines". In: *2003 IEEE XIII Workshop on Neural Networks for Signal Processing (IEEE Cat. No.03TH8718)*. IEEE. DOI: 10.1109/nnsp.2003.1318056. URL: https://ieeexplore.ieee.org/abstract/document/1318056.

Ravanbakhsh, Mahdyar et al. (June 23, 2017). "Training Adversarial Discriminators for Cross-channel Abnormal Event Detection in Crowds". In: arXiv: 1706.07680 [cs.CV]. URL: https://arxiv.org/abs/1706.07680.

Sakurada, Mayu and Takehisa Yairi (2014). "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction". In: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis - MLSDA'14*. ACM Press. DOI: 10.1145/2689746.2689747.

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

Schlegl, Thomas et al. (Mar. 17, 2017). "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery". In: *Lecture Notes in Computer Science*. Springer International Publishing, pp. 146–157. DOI: 10.1007/978-3-319-59050-9_12. arXiv: 1703.05921 [cs.CV].

Schölkopf, Bernhard et al. (Jan. 1999). "Support Vector Method for Novelty Detection". In: vol. 12, pp. 582–588. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.575&rep=rep1&type=pdf.

Yu, Qien, Muthu Subash Kavitha, and Takio Kurita (2020). "Mixture of experts with convolutional and variational autoencoders for anomaly detection". In: *Applied Intelligence*. DOI: 10.1007/s10489-020-01944-5. URL: https://doi.org/10.1007/s10489-020-01944-5.

Zenati, Houssam et al. (Feb. 17, 2018). "Efficient GAN-Based Anomaly Detection". In: arXiv: 1802.06222 [cs.LG].