POLITECNICO
MILANO 1863

Department of
Electronics, Informatics and Bioengineering
M.Sc. course of Computer Science and Engineering

Adversarially Learned Anomaly Detection Using Generative Adversarial Networks

Supervisor:
Doc. Giacamo Boracchi — Politecnico di Milano
Co-Supervisors:
Diego Carrera — Politecnico di Milano

Master of Science Thesis of:
Şemsi Yiğit Özgümüş
Matriculation ID 893558

July 2019

To my fiancé and my family

Acknowledgements

Contents

List of Figures

Page

List of Tables

Page

Abstract

Anomaly detection is an increasingly popular field in computer vision and is a challenging problem concerning various application domains. As the methods of gathering and utilizing data grows, quality and the performance of the systems we benefit in our daily lives become more dependent on the data. Considering these systems relies on the quality of data, anomaly detection plays a crucial role to detect abnormalities that may harm the system if left undiscovered. In scenarios where the abnormal data is a rare occurrence or simply costly to be observed, reconstruction based approaches are used to model the normal data to learn to differentiate anomalies when encountered. Generative adversarial networks is relatively new method that is used to generate new and unobserved data in an unsupervised way and its adoption to reconstruction based anomaly detection framework is promising.

This thesis provides a model to predict the anomalous regions on the Scanning Electron Microscope (SEM) images of nanofibrous materials using a framework based on generative adversarial networks (GAN) and encoder networks. Proposed solution learns the bidirectional mapping between the image and its latent dimension space using the adversarial training of the generator of generative adversarial network and the encoder network.

We show that our approach produces better results than other GAN based anomaly detection frameworks trained with our dataset. While the performance increase is a small margin, our method shows better visual reconstructions of the data.

Sommario

L'individuazione di anomalie è un campo sempre più popolare nella visione artificiale ed è una sfida problema riguardante vari domini applicativi. Come i metodi di raccolta e utilizzo i dati crescono, la qualità e le prestazioni dei sistemi di cui beneficiamo nella nostra vita quotidiana diventano più dipendente dai dati. Considerando questi sistemi si basa sulla qualità dei dati, il rilevamento delle anomalie svolge un ruolo cruciale per rilevare anomalie che potrebbero danneggiare il sistema se non vengono scoperte. In scenari dove dati anormali sono eventi rari o semplicemente costosi da osservare, approcci basati sulla ricostruzione sono usati per modellare i dati normali per imparare a distinguere le anomalie quando incontrate. Contraddittorio generativo le reti sono un metodo relativamente nuovo che viene utilizzato per generare dati nuovi e non osservati in modo non controllato e la sua adozione alla ricostruzione basata sul quadro di rilevamento delle anomalie è promettente.

Questa tesi fornisce un modello per prevedere le regioni anomale sulle immagini del microscopio elettronico a scansione (SEM) di materiali nanofibrati utilizzando un framework basato su reti generative adversarial (GAN) e reti di encoder. La soluzione proposta apprende la mappatura bidirezionale tra l'immagine e la sua dimensione latente usando il contraddittorio formazione del generatore della rete generativa avversaria e della rete di encoder.

Dimostriamo che il nostro approccio produce risultati migliori rispetto ad altri framework di rilevamento delle anomalie basati su GAN formati con il nostro set di dati. Mentre l'aumento delle prestazioni è un piccolo margine, il nostro metodo mostra una migliore ricostruzione visiva dei dati.

CHAPTER 1

Introduction

Introduction will be written

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fer-

mentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

This document is organized as follows:

Chapter 2 outlines the state of the art methods that is used with this type of dataset. Chapter 3 explains other anomaly detection methods that benefit from the generative adversarial networks. Chapter 4 presents the anomaly detection problem throughly and reconstruction based approaches and the proposed approach for the problem. Experimental results are described in Chapter 5 and the conclusion and the future work is stated in chapter 6.

CHAPTER 2

State of the Art

This chapter is dedicated to the exploration of state of the art approaches in the fields relevant to the thesis. The organization of the chapter is the following: Section 2.1 will formulate anomaly detection problem and illustrate mainstream approaches towards its solution. Section 2.2 will give necessary background about generative adversarial networks (GAN) and autoencoder networks. First, the principles behind how GANs work will be presented then the Autoencoder Networks will be explained. This section is particularly significant to understand how different architectures that use these networks actually work. In Section 2.3, variety of architectures that has been developed for anomaly detection will be studied as state of the art. How they are implemented and their purpose will be described.

2.1 Anomaly Detection

Anomaly detection can be defined as the task of identifying test data that differs in some respect from the data for the model that is available during training [45]. This definition correctly describes the task on the abstract level but does not clarify what an anomaly actually is. Depending on the the application domain, anomalies can be referred to as outliers, novelties, discordant observations, exceptions, aberrations, surprises, peculiarities and contaminants [15]. Sometimes these terms can be used interchangeably in the context. These distinctions in the definition primarily rely not only on the features of the data such as its dimensional complexity, its continuity and format, but also on what is considered an anomaly for that specific domain. This makes it hard to conceptualize the prob-

lem of anomaly detection with a general definition and a solution that can encompass all different domains.

In the rest of this section, I will explain different use cases for anomaly detection to show its impact on a broad spectrum of problems. Then the primary approach types to solve anomaly detection problem will be introduced.

The categorization taxonomy to define different application domains is constantly changing parallel to developments in the literature but the application areas of anomaly detection can be categorized down to these five general fields [45]. These are:

- IT Sector
- Healthcare informatics, medical diagnostics and monitoring
- Industrial Monitoring/ maintanence and damage detection
- Image processing/ Video Survailance
- Survailance
- Sensor Networks

IT Sector

Anomaly detection methods have a huge impact on the IT security systems, in which the objectives include network intrusion detection and fraud detection [45]. With rising use of computerized systems and the incorporation of the internet based services to our daily lives, maintainability and the availability of these systems became much more important. For systems like these, anomaly is considered as a priority because it indicates significant but rare events and can prompt to critical actions to be taken in a domains such as finance or cyber security [1]. For example a malicious process in a server may show increased use of resource from time to time that may be spotted as a type of anomaly in the system resource use patterns or the unusual amount of traffic on the network may indicate an intruder attack [20] [28]. Another aspect of the increased use of internet is its impact on the e-commerce field and dramatically growing online payment systems. This also includes a migration to internet banking in favor of technological convenience. As a result, insurance and credit card frauds, scams towards phone and mobile data usage, and phishing attacks targeting financial transactions become more and more severe [21]. In this context, behavior of the consumer becomes the data. Transaction history, time spent between consecutive transactions and the geographic data can be used to construct a model to detect anomalies in the purchase patterns.

Healthcare

Healthcare and medical diagnostics is another domain for anomaly detection. Detecting anomalies in the progression of the disease symptoms is a crucial factor for treatment monitoring [55]. To aid diagnostics, interpretation of the patient data is also an important task for anomaly detection. Ability to differentiate between the instrumentation or recording errors and the actual clinically relevant change that may indicate a symptom should also be present in these anomaly detection systems. [45]. Patient data may include tabular information like the age, weight, blood test results, heart or respiratory rate [16][39] or medical image data [55].

Industrial Monitoring

Novelty detection became a huge part of the industrial automation systems from the quality control of the manufacturing progress to the performance maintenance of the equipment. Products that have a complicated and delicate manufacturing process like CPU wafers [30] and nanofibrous materials [42] may easily develop defects in the manufacturing pipeline. Using anomaly detection is a viable option to reduce the wasted material and optimize the cost of producing the target product. Same principle also applies to the equipment used in these pipelines. Equipment and assets have the risk of deteriorating over time due to the usage and wear [45]. For example heavy extraction machines that is used in the petroleum industry like turbo machines [40] needs constant supervision for damage prevention.

Image Processing/ Video Surveillance

Image processsing methods has been used with anomaly detection methods extensively to detect novelty subjects or anomalies in the image or in a video stream. Detection in video data stream data is an important task for the surveilance systems that is used with the purpose of traffic control, security systems and search and rescue [5].

Sensor Networks

With the improvements of smart devices' involvement in our daily lives and the recent but promising developments in the field of internet of things, the definition of acquiring and processing data is evolved. Various types of sensors, especially wireless, are heavily used to collect huge amount of data to feed various systems such as surveillance, environmental monitoring, and disaster management [59]. Anomaly detection methods are important for these systems because the performance and maintenance of the these system requires constant monitoring of the acquired data. These distributed systems are mainly used to collect and compute the gathered sensory data and predict target information. However the accuracy of these computations or

predictions depends primarily on the quality of the sensor measurements and with this scale the instrumentation error is unavoidable. That's why anomaly detection tasks usually deployed on these systems to observe sensory faults and also prevent errors caused by malicious attacks [45] [9].

Variety of approaches that belong to different sub domains of the machine learning, statistics , bayesian and information theory are used to solve anomaly detection problem for these aforementioned fields. All these different approaches in general tries to build a model or a representation of the data that is provided which contains no anomalous samples (or very few) and assigns them an anomaly score or a label depending on the problem. At the inference stage, deviations from the constructed normality definition are detected using a predetermined anomaly threshold. These different methods can be categorized into 5 main titles. These are:

- Probabilistic Anomaly Detection Methods
- Distance Based Anomaly Detection Methods
- Domain Based Anomaly Detection Methods
- Information Theoretic Anomaly Detection Methods
- Reconstruction Based Anomaly Detection Methods

Probabilistic methods aim to estimate the generative probability density function of the data with the assumption that the input data is generated from an unknown probability distribution D, and learning this distribution provides us a mathematical representation of normality for the input data, training set. Anomaly (or novelty in some approaches) score then can be determined which has certain probabilistic interpretation and can be used to detect the anomaly on the inference stage [45]. **Distance based methods** are another type of anomaly detection technique used to model the training data. Including the K-Nearest neighbour based and clustering based methods, distance based methods rely on distance metrics between the data points to create a representation for normality. For example considering the K-NN approach, data points in the training dataset are identified to have close neighbour points using the distance metrics while a sample that has greater distance to the nearest normal point is identified as an anomaly. **Domain based methods** creates a boundry based on the structure of the training dataset [45]. These methods are usually insensitive to different sampling or the density properties because instead of learning the density of the data, the learn the boundary information to seperate the target class (anomalies) from the normal data. Support Vector Machines (SVM) are the popular example for this approach. **Information theoretic anomaly detection methods** computes the total information content that

resides in the training dataset. The main assumption is that anomalous samples present in the data significantly modifies the information content of the assumed normal dataset. Metrics for this approach are calculated using the entirety of the dataset and the subset that triggers the biggest discrapancy is identified as the anomaly.

Reconstruction based approaches which also is the focus of this thesis, model the underlying representation of the data and reconstructs it. Using the original input and the reconstructed output, these models define a reconstruction error tailored to the nature of the problem and the type of data, and measures this error with respect to certain anomaly threshold. Since the modeled underlying data has the features specific to the training dataset, when the anomalous sample is fed through this pipeline, its representation and consequently its reconstruction will be made poorly hence giving a greater reconstruction error.

2.1.1 Related Works

In this subsection, previous works and state of the art related to anomaly detection in images and proceed with particular case of SEM images (see section 4.1) are presented. Proposed methods for anomaly detection for images can be grouped in reference-based and reference-free ones [15]. Reference-based approaches compares the input image with a template image which is anomaly-free by definition and result of this comparison decides in run-time whether input includes anomalous component or not [65]. However, they are not feasible for considered SEM images where filament structures follow randomized rather than geometric pattern [13]. Considering this distinction, our proposed model is a member of reference-free approach which learns to model the data with an unsupervised setting.

This problem of particular dataset of SEM images previously studied in [13], [14] and [12] and these methods are based on learning a sparse dictionary that represents the normal regions and in run-time, if the input does not conform with the learned dictionary, it is detected as anomalous. Another approach that introduces convolutional neural networks to anomaly detection task in SEM images is established by Napoletano et al. [42]. To extract feature for "normality" concept, they exploit pretrained network called ResNet [26] which consists of residual networks and trained by large number of images. After feature extraction, for test phase, they create a dictionary by using K-means clustering from normal subregions in the training data and use this to compare with test samples to determine the degree of abnormality [42].

Another powerful model for novelty detection is autoencoders (see Section 2.2.2). Autoencoders are quite useful tools to learn the dynamics of a distribution defined over images and to generate examples that corresponds to a sample on learned distribution. Therefore, they are easily deployed for novelty detection as in [4], [36] and [44]. Particularly, in Sabokrou et al.[50], leverages two staged network that is composed by auto-encoders and convolutional neural networks is explored for the novelty detection.

Next section will introduce generative adversarial networks and autoencoder networks to give background for the GAN based anomaly detection frameworks introduced in Section 2.3.In the GAN section, Underlying mechanics of adversarial training and how generative adversarial networks learn to generate given data will be explained.Regarding autoencoder networks, main principle behind their implementation and how they are trained to obtain a latent representation of a given data will be described.

2.2 Architectural Components

This section introduces the two main network types used in anomaly detection frameworks.Section 2.2.1 will describe how generative adversarial network framework is formed and the intuition behind its adversarial training scheme. Section 2.2.2 will give an introductory information to autoencoder networks and present different kinds of approaches that can be integrated into both the construction of the network and its training methodology.

2.2.1 Generative Adversarial Networks

One of the main objectives of the deep learning is to create complex and high dimensional models that can capture the relations amongst the fundamental types of data that we work with. This data may differ, it can be a visual information like an image, symbols in natural language or audio waveforms containing speech. We then use these models to make machines make sense of the world we live in and exhibit what we call an intelligence [10].

In the terms of statistical classification, also in machine learning, two main approaches to build a classifier are called discriminative and generative modelling. They differ in the degree of statistical modelling[29]. Discriminative models maps the rich, complex sensory data to a class label to perform the desired task. They try to learn the conditional probability

distribution of the input data using the features of the data. Success behind this approach is the result of the improvements in the training, learning and generalization of these models. Backpropagation [60], dropout [58], and rectified linear units[22] are some of the example concepts that contribute to this success.

Generative models try to estimate the joint probability distribution of the high dimensional sensory input data to be able to generate data similar to input and its label with some variations which can be thought as imaginative. Models such as Boltzmann machines [52] and Gaussian Mixture Models [47] did not show a considerable promise as opposed to discriminative models mainly because of the type of problem they try to solve and computational difficulties regards to that solution. It is observed that main difficulty was the probabilistic computations that are hard to control in the maximum likelihood estimation [24, 51].

Generative Adversarial Networks (GAN)[24] are considered as one of the most significant breakthroughs in deep learning and generative modelling in the recent years. Framework consists of a generator and discriminator network. You can observe the example architecture in the figure 2.1.

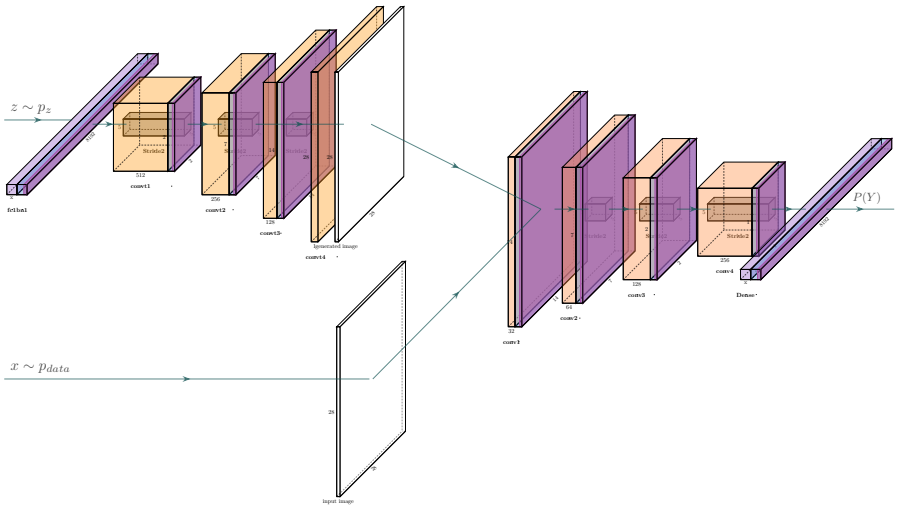


Figure 2.1 – GAN architecture overview

From this point on, "image" will be used to specify the type of the input and the target data because GAN frameworks are mainly designed to work with high dimensional spatial data like images so it will be consistent when explaining the theorem and also the central focus of the thesis is on the spatial data.

The power of this framework comes from the adversarial nature of the training process. Simply put, the goal of generator network(G) is to create images similar to the input data by capturing the data distribution using a probabilistic model. On the other hand discriminator (D) tries to detect fake (generated) images that is created by generator from the real input images. We can represent this intuition with the following two player minimax game with the value function $V(G, D)$ in Equation 2.3. Here P_{data} represents the probability distribution of the data that we have no information about. $p_z(z)$ is the random noise that generator accepts to generate fake data. [24]

$$p_{\text{data}}(x) : x \in \Omega_X \quad \text{(Distribution of the data)} \tag{2.1}$$

$$p_z(z) : z \in \Omega_Z \quad \text{(Distribution of the noise)} \tag{2.2}$$

$$V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{ data }}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.3}$$

The training objective for the discriminator is to maximize the expected log-likelihood of classifying the input data as real. Since generated data should be classified as fake by the discriminator, the output of the second term is flipped. The Generator's purpose is to "fool the discriminator". Hence its objective is to minimize the expected log likelihood of discriminator classifying generated image sample as fake. We can interpret this equation as two seperate objective functions with different training criterions :

$$\min_G V(G, D) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.4}$$

$$\max_D V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{ data }}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.5}$$

While this equation describes the relation between two adversaries, it does not provide a definite explanation how this training objective finds equilibrium and whether that equilibrium also provides us with the optimal generator or not. For this analysis we will use bottom up approach and prove the theorem depicted in the original GAN method [24].

Theorem 2.2.1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-log(4)$.*

We start by defining the term expected value of a function.

Definition 2.2.1. $E(f(x))$ of some function $f(x)$ with respect to certain probability distribution $p(x)$ is called the expected value of a function.

It is expressed as:

$$E_{x\sim p_x} = \int p(x)f(x)dx \tag{2.6}$$

So the minimax game equation can be expressed like this:

$$V(G,D) = \int_x p_{\text{data}}(\boldsymbol{x})log(D(\boldsymbol{x}))dx + \int_z p_z(\boldsymbol{z})log(1-D(G(\boldsymbol{z})))dz \tag{2.7}$$

For generator to sample an image good enough to fool the discriminator, framework needs to learn the implicitly created sample distribution p_g by the generator G. Framework defines a prior noise distribution $z \sim p_z$, isotropic Gaussian distribution with zero mean and unit variance to learn the distribution p_g (Different distributions are also considered). Conceptually p_z represents the latent features of the data artificially created. While training GAN, semantic meaning of this noise is not controlled. Next we use the LOTUS theorem.

Theorem 2.2.2. *Law of the unconscious statistician (LOTUS) theorem is used to compute the expected value of a function $g(x)$ of a random variable X when one knows the probability distribution of X but does not know the probability distribution of $g(x)$. [48]*

The expected value of a function $g(x)$ then can be expressed like:

$$\mathrm{E}[g(X)] = \sum_x g(x)f_X(x) \tag{2.8}$$

$f_X(x)$ being the probability distribution of the random variable X.

Using this law we can write the equation $V(G,D)$.

$$z \sim p_z, \quad G(z) = x, \quad x \sim p_g \tag{2.9}$$

$$V(G,D) = \int_x p_{\text{data}}(\boldsymbol{x})log(D(\boldsymbol{x})) + p_g(\boldsymbol{x})log(1-D(\boldsymbol{x}))dx \tag{2.10}$$

For the Discriminator D, the value of the training criterion should be maximized. That basically means the function derivative of this expected value expression should be equal to zero. We write the function by reversing

the process of the equation 2.6.

$$f(p_{\text{data}}, p_g) = p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x})) \tag{2.11}$$

$$f' = p_{\text{data}}(\boldsymbol{x}) \frac{1}{D(\boldsymbol{x}) \ln(C)} - p_g(\boldsymbol{x}) \frac{1}{(1 - D(\boldsymbol{x})) \ln(C)} \tag{2.12}$$

$$f' = 0 \tag{2.13}$$

$$p_{\text{data}}(\boldsymbol{x})(1 - D(\boldsymbol{x})) \ln(C) = p_g(\boldsymbol{x}) D(\boldsymbol{x}) \ln(C) \tag{2.14}$$

$$(p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})) D(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x}) \tag{2.15}$$

$$D^*_G(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{(p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x}))} \tag{2.16}$$

This is the optimal discriminator D with the fixed G. By deriving this value, we show that the maximum value the Discriminator can reach is the result of the equation 2.16. Next, we reformulate our $V(G, D)$ to analyze for the training criterion of the generator G.

$$C(G) = \max_D V(G, D) \tag{2.17}$$

$$= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [\log (1 - D^*_G(G(\boldsymbol{z})))] \tag{2.18}$$

$$= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [\log (1 - D^*_G(\boldsymbol{x}))] \tag{2.19}$$

$$= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right] + \mathbb{E}_{\boldsymbol{x} \sim p_g} \left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right] \tag{2.20}$$

In the theorem, equality $p_{\text{data}} = p_g$ is given. By applying this equality we get $D^*_G(\boldsymbol{x}) = \frac{1}{2}$. By putting this to our equation we get

$$C(G) = \log(\frac{1}{2}) + \log(\frac{1}{2}) = -\log(4)$$

. To verify that this value is indeed the global point for the equation we apply the following subtraction:

$$
\begin{aligned}
C(G) - (-\log(4)) &= \int p_{\text{data}} \left[\log \frac{p_{\text{data}}}{p_{\text{data}} + p_g} + \log 2 \right] + \\
&\quad \int p_g \left[\log \frac{p_g}{p_{\text{data}} + p_g} + \log 2 \right] \\
C(G) - (-\log(4)) &= \int p_{\text{data}} \left[\log \frac{2p_{\text{data}}}{p_{\text{data}} + p_g} \right] + \int p_g \left[\log \frac{2p_g}{p_{\text{data}} + p_g} \right] \\
C(G) - (-\log(4)) &= \int p_{\text{data}} \left[\log \frac{p_{\text{data}}}{(p_{\text{data}} + p_g)/2} \right] + \int p_g \left[\log \frac{p_g}{(p_{\text{data}} + p_g)/2} \right]
\end{aligned} \tag{2.21}
$$

To interpret this result we introduce 2 divergence definitions.

Definition 2.2.2. The Kullback-Leibler divergence is the measure of how one probability distribution is different from the other one.

Its general definition is:

$$
D_{KL}(P \| Q) = \int p(x) \log \frac{p(x)}{q(x)} \tag{2.22}
$$

More generally if P and Q are probability measures[1]over a dataset X, and measure P is absolutely continous with respect the Q, then the Kullback-Leibler divergence can also be defined as in equation 2.23

$$
D_{\text{KL}}(P \| Q) = \int_{\mathcal{X}} \log \left(\frac{dP}{dQ} \right) dP \tag{2.23}
$$

where $\frac{dP}{dQ}(f_{PQ})$ is the Radon-Nikodym derivative [11]. The defining property of Radon-Nikodym property that ties to the probability measure is that:

$$
P(R) = \int_R f_{PQ} dQ \tag{2.24}
$$

The RN derivative $f_{PQ} : \Omega \mapsto \mathcal{R}_{\geq 0}$ is defined for any measures P and Q on a space Ω such that P is absolutely continous with respect to Q. In other words, for any $R \subseteq \Omega : P(R) > 0 \implies Q(R) > 0$. [11]

[1]A probability measure is a real-valued function defined on a set of events in a probability space that satisfies measure properties such as *countable additivity*[49]

Jensen-Shannon divergence is another measure of similarity between different probability distributions. Its major difference from the KL divergence is that it is always symmetric and it always has a non-negative value. The General definition is provided in equation 2.25.

$$JSD(P\|Q) = \frac{1}{2}D_{KL}(P\|M) + \frac{1}{2}D_{KL}(Q\|M) \tag{2.25}$$

Using this divergence measures we can rewrite our equation as this:

$$C(G) = -\log(4) + KL\left(p_{\text{ data}} \| \frac{p_{\text{ data}} + p_g}{2}\right) + KL\left(p_g \| \frac{p_{\text{ data}} + p_g}{2}\right) \tag{2.26}$$

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{ data}} \| p_g\right) \tag{2.27}$$

Considering Jensen-Shannon divergence is always non-negative and zero only when the distributions are identical, we can conclude the proof by stating that the global minimum point of $C^*(G) = -\log(4)$ is only achieved when $p_{\text{data}} = p_g$.

Intuition behind the adversarial training of this framework is important because all the subsequent approaches that will be discussed in the thesis that use GAN's fundamentally benefit from this framework to train their networks. As we will see in the BiGAN (section 2.3.2) and similar models, addition of encoder network changes the overall minimax game played by the adversary networks but underlying logic still depends on the value function of the GANs.

2.2.2 Autoencoder Networks

Autoencoder network is a type of neural network that is trained to copy its input data to output. [23] In its simplest form, it can be defined as a feedforward, non- recurrent network similar to multilayer perceptrons like in figure 2.2. Internally, it consists of a hidden layer that describes the representation (code) learned from the input data. Connections from the hidden layer to the output layer transforms the hidden representation into the output data with the same dimensionality as the input. Rather that trying to reconstruct the given data, autoencoders are mainly used to learn the representation about the data to extract meaningful features and to reduce the dimensionality of data while keeping the relevant information as an alternative to principal component analysis.

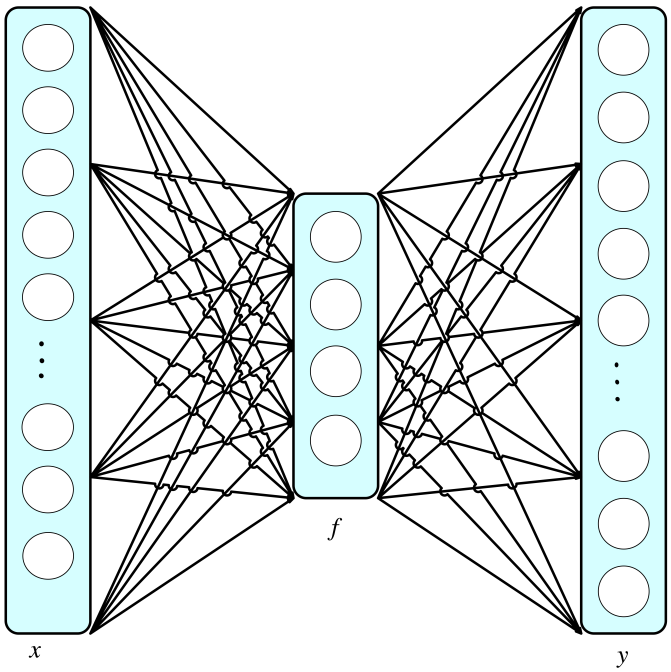


Figure 2.2 – Simple Autoencoder architecture

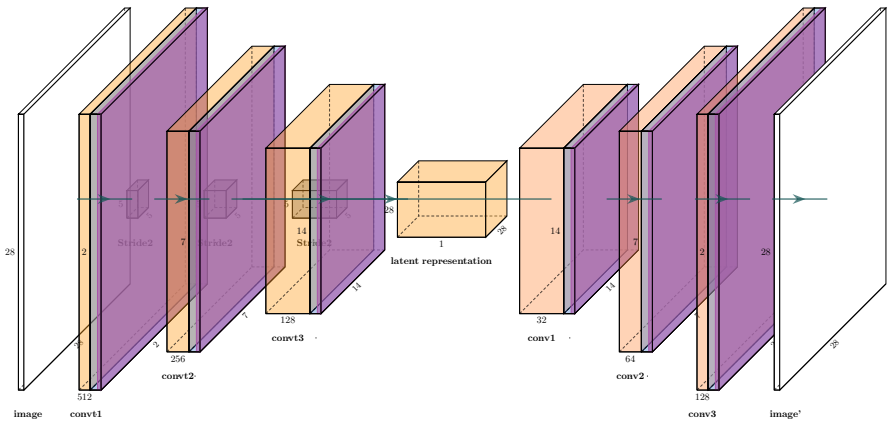


Figure 2.3 – Stacked Autoencoder with multiple layers

With the increased complexity of the data present, this architecture can be extended to capture more complex feature representations by stacking similar but varying encoder and decoder layers on top of each other.

Generally first layers learn the more basic features of the data and as the layers are stacked, more concentrated and complex features can be learnt in the code section of the autoencoder, end of the encoder network. These architectures are mainly referred to as deep autoencoders because of the increased level of hidden layers present in the model depicted in figure 2.3, just like in deep learning.

Learning rationale of the autoencoders can be visualized with a pair of weighted layer computation functions like in the multilayer perceptrons. Assume that we have two transition functions that defines the relationship between the real data and the intermediate representation as the result of the encoder network:

$$\phi : X \mapsto F$$
$$\psi : F \mapsto X$$

We can define the intermediate representation computation and the reconstruction of input data with the following equations:

$$z = \sigma(Wx + b), \quad x \sim X \in \mathbb{R}^d, \quad z \sim F \in \mathbb{R}^p \tag{2.28}$$
$$x' = \sigma'(W'z + b') \tag{2.29}$$

The primal training objective for the autoencoders is the reconstruction loss. Depending on the architecture of the decoder or the desired task, the source of parameters for the objective function can change. Autoencoder learns the representation of data by minimizing the objective function defined as the reconstruction loss.

$$\phi, \psi = \underset{\phi,\psi}{\operatorname{argmin}} \|X - (\psi \circ \phi)X\| \tag{2.30}$$
$$L(x, x') = \left\|x - x'\right\|^2 = \left\|x - \sigma'(W'\sigma(Wx + b) + b')\right\|^2 \tag{2.31}$$

With enough model capacity and training time, autoencoder might learn to copy the input perfectly. This reduces the functionality of the encoder-decoder modules to a identity function:

$$x \sim X \in \mathbb{R}^d, \quad \psi(\phi(x)) = x$$

This is generally an unwanted outcome. To overcome this problem, autoencoders are designed to learn the representation of the data imperfectly.

[23] This is achieved by how the encoder module learns to compress the data to a meaningful representation. These 3 types of autoencoder models can give an idea about how the architecture can be changed to alter the amount of information that can be extracted from the data. These are:

- Undercomplete Autoencoders
- Regularized Autoencoders
- Denoising Autoencoders

Undercomplete autoencoders reduce the dimension size of the latent representation. Learning purposefully insufficient representation forces autoencoder to keep the most salient features about the data. Training objective function is the reconstruction loss which presented in equation 2.30. If the computation of this function is linear, then the autoencoder mimics the behaviour of a principal component analysis method. When the encoder transition ϕ and the decoder transition ψ is designed to have a nonlinear nature, then the autoencoder has the ability to capture more complex reprensetation of the data, more powerful approach than known dimensionality reduction methods such as Principal Component Analysis. But this has the risk of again, causing autoencoder to replicate the input data perfectly and becoming an identity function $x' = \psi(\phi(x))$. Regularized autoencoders' design difference is that their latent representation layer usually has same or higher dimensional power (more neurons from the perspective of multilayer perceptrons). To acquire a similar performance like undercomplete variants, they regularize the loss function with the potential properties of the data distribution such as robustness to the noise or the sparsity of the representation. Prioritizing which aspects of the data will be used as a regularizer term for the objective function is a task dependent problem. [23]. Denoising autoencoders tries to solve a different problem to achieve a good representation.

$$\hat{x} = x + \epsilon : x \sim X \in \mathbb{R}^d, \quad \epsilon \sim \mathcal{N}(\mu, \sigma^2)$$

The input data is corrupted with some kind of noise, gaussian etc. Denoising autoencoders tries to reconstruct the data and undo this corruption by minimizing this updated loss with the difference of $\hat{x}$

$$L(x, \psi(\phi(\hat{x})))$$

In the remaining part of this thesis, these networks will be referenced several times. They constitute the building blocks of all the models presented in the next section. Our proposed model's architecture also struc-

tured using these networks so their underlying logic is fundamental to understand to compose our solution.

2.3 GAN Based Anomaly Detection Methods

In the field of reconstruction based anomaly detection, autoencoder architectures are fairly popular with spatial data [8, 36, 4]. GAN based approaches in this field are relatively new compared to their other variants. In the following subsections, we will explain the approaches that can be considered as the state of the art for this problem in the order of their complexity and incremental improvements. Each approach will be evaluated by their problem description, the type of data they use, model architecture and anomaly score computation.

2.3.1 AnoGAN

AnoGAN model is the first approach that utilizes GANs for the anomaly detection task. [55] It focuses on the problem of detecting anomalous regions in optical coherence tomography images. Images containing retinal fluid or hyperreflective foci (indicator of disease progression in various retinal diseases) are identified as anomalous examples. The main motivation to use GANs to obtain a model which represents normal anatomical variability of the data is that previous supervised approaches involves training a model with a large dataset using annotated examples of known markers. Relying on vocabulary of markers may limit the predictive power of the model as the image contains much more relevant information than the marker and extensive supervised training is a problem because it means that every stage of the disease requires an extensive training with annotated examples such as labeled lesions which may decrease limit of exploiting imaging data for treatment decisions. [55].

The framework consists of 2 parts. First part is the GAN module, generator and discriminator networks (see figure 2.1). Their architecture is inspired by the original DCGAN [46] which contains convolutional layers instead of multi layer perceptron layers. The objective function of the GAN is the same of the equation 2.3. During the training the generator enhances its ability to generate more realistic retinal images while the discriminator learns the diagnose the real and the generated sample more efficiently.

The second part of the framework is about the inverse mapping of the noise to the image. We know from the GANs training procedure that when

the adversarial training is completed, the generator network has learned the distribution p_g that is theoretically very similar to the input data distribution p_{data}. (I state "theoretically" explicitly because depending on the configuration of GAN and the data that is used, this may differ significantly in practice. This aspect of the learning will be mentioned in the latter approaches.) Even though with the trained generator,

$$G(z) = z \mapsto x, x \sim p_{\text{data}}$$

the GAN framework does not learn the inverse mapping from the image to noise. AnoGAN framework aims to find the noise sample $z\prime$ from the query image (normal or anomalous) such that the generated image $G(z\prime)$ is visually the most similar one to the query image. This is essential for the anomaly detection task. $z\prime$ is sampled randomly as initialization. Then the $G(z\prime)$ is computed and based on the defined loss function gradients the coefficients of $z\prime$ is updated using back propagation. This procedure is repeated for a number of epochs to obtain the final value of the $z\prime$.

The loss function for the inverse mapping is defined as the combination of two separate losses. Residual loss is responsible for enforcing visual similarity between the generated image $G(z\prime)$ and the query image x.

$$L_R(z_\gamma) = \sum \|x - G(z_\gamma)\| \tag{2.32}$$

Discrimination loss on the other hand enforces the generated image $G(z\prime)$ to lie on the manifold X of image x.

$$L_D(z_\gamma) = \sum \|f(x) - f(G(z_\gamma)\| \tag{2.33}$$

There are two different implementation for the discrimination loss. One with sigmoid cross entropy which is from the original GAN implementation [24] and one with feature matching [53] which uses the last layer of the discriminator rather than the output to compute the loss. Using this loss allows them to use discriminator not as a classifier but as a feature extractor. Using both residual and discrimination loss, the total loss function is defined as :

$$L(z_\gamma) = (1 - \lambda) \times L_R(\boldsymbol{z_\theta}) + \lambda \times L_D(\boldsymbol{z_\gamma})$$

Only the coefficients of z is adapted via back propagation. The trained weights of the generator and discriminator are fixed in this stage. [55]

Anomaly detection task uses same loss functions to compute the anomaly

score.

$$A(x) = (1 - \lambda) \times R(\boldsymbol{x}) + \lambda \times D(\boldsymbol{x})$$

The $R(x)$ is the residual score value and the $D(x)$ is the discrimination score value at the last update iteration of the inverse mapping procedure, respectively. The model outputs a larger anomaly score $A(x)$ for anomalous samples though smaller score means that very similar image is seen during the training of GAN.

This model is the first model that integrates generative adversarial networks to an anomaly detection framework. The mapping from latent representation to image data is completely dependent on the architecture of GAN and its training parameters. The disadvantageous aspect of this approach is the inference. For each query image framework approximates the latent representation vector using back propagation steps which makes the whole process significantly slow compared to the other approaches. Improvements to this framework is discussed in the Latest Developments chapter (3.1). The experiment results and the implementation details will be covered in chapter 5.

2.3.2 BiGAN and ALI

BiGAN (Bidirectional Generative adversarial Networks) [18] and ALI (Adversarially Learned Inference) [19] are two models that independent from each other but focuses on the same problem. They investigate to acquire efficient inference method from data to latent space by using encoders in the adversarial training framework of GANs. They emphasize learning the inverse mapping for the auxiliary supervised tasks such as classification, segmentation and discrimination. The only difference between the models is that the BiGAN uses a deterministic network for the encoder while ALI uses a stochastic network. I will first briefly explain the original architecture of the ALI and continue to present the framework in detail using BiGAN instead since their explanation for the model is more consistent with the GAN framework. Furthermore I will discuss how this architecture can be used for anomaly detection. In the original papers, anomaly detection has not been explored as an auxilary supervised task but the later models like in section 2.3.3 or in section 2.3.4, used these architectures as a state of the art comparison in terms of both foundation and performance measure hence their use of anomaly detection will also be discussed.

Aligan architecture is depicted as a generator and a discriminator in figure 2.4. The significant change in ALI is that the generator is the com-

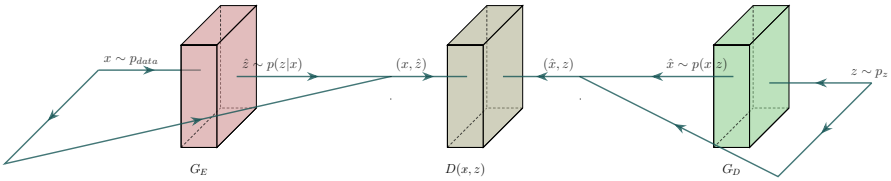


Figure 2.4 – ALI architecture overview

bination of decoder network and an encoder network. Encoder represents the joint distribution $p_E(x, z)$ which has the marginal $p_{\text{data}(x)}$ and decoder represents the joint distribution $p_D(x, z)$ with the marginal $p_z(z)$ which represents the factorized noise distribution (generally $\mathcal{N}(0, 1)$). The objective function of the ALI is to match these two joint distributions. Successing in this goal also ensures the match of marginals and conditional distributions $p(z|x)$ and $p(x|z)$. Discriminator is also modified to support this new objective function. It is trained to distinguish between the joint pairs $(x, \hat{z} = G_E(x))$ and $(\hat{x} = G_D(z), z)$ as data and noise.

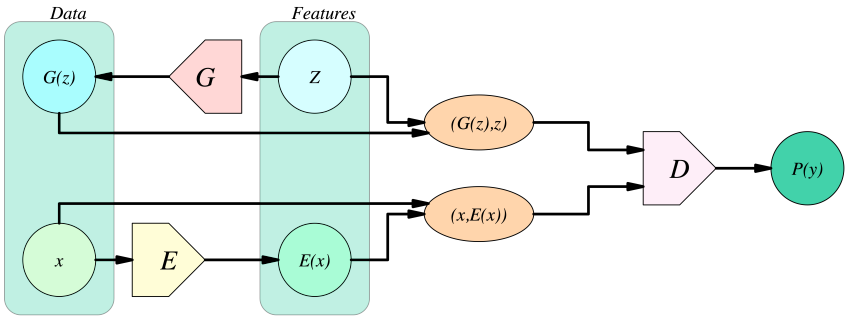


Figure 2.5 – BiGAN architecture overview

BiGAN architecture model is presented in figure 2.5. The architectures are theoretically identical with the exception of the encoder network choice in practise. BiGAN experiments are conducted with a deterministic network whilst ALI framework used a stochastic network. Apart from the generator, BiGAN also trains the encoder network in an adversarial manner. Same objective function used in GAN provides the relationship between the generator and encoder that $G = E^{-1}$ and vice versa. This invertibility is the key aspect of the model and other frameworks to provide an efficient inference with reliable sampling. We Know from the section 2.2.1 that:

$$G : \Omega_z \mapsto \Omega_x$$

We define the Encoder with reverse of that transition:

$$E : \Omega_x \mapsto \Omega_z$$

Since this framework accepts the networks as deterministic we also redefine the distributions for generator and encoder networks:

$$p_G(x|z) = \delta(x - G(z))$$
$$p_E(z|x) = \delta(z - E(x))$$

The objective function of the framework can be characterized as:

$$\min_{G,E} \max_{D} V(D, E, G) \tag{2.34}$$

where

$$V(D,E,G) := \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{X}}} \underbrace{\left[\mathbb{E}_{\mathbf{z}\sim p_E(\cdot|\mathbf{x})}[\log D(\mathbf{x},\mathbf{z})]\right]}_{\log D(\mathbf{x},E(\mathbf{x}))} + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{Z}}} \underbrace{\left[\mathbb{E}_{\mathbf{x}\sim p_G(\cdot|\mathbf{z})}[\log(1 - D(\mathbf{x},\mathbf{z}))]\right]}_{\log(1-D(G(\mathbf{z}),\mathbf{z}))} \tag{2.35}$$

To prove that the encoder is the inverse of the generator, optimal discriminator needs to be determined.[18] The joint distributions of the generator and encoder can be illustrated as the marginals of the data and noise and related conditional probability distributions:

$$p_{GZ}(x, z) := p_G(x|z)p_Z(z) \tag{2.36}$$
$$p_{EX}(x, z) := p_E(z|x)p_X(x) \tag{2.37}$$

The joint latent space and the data for the problem can also be illustrated as a joint representation.

$$\Omega := \Omega_X \times \Omega_Z$$

For the region $R \subseteq \Omega$ probability measures[32] of encoder and generator is defined in equation 2.38 and 2.41 respectively.[18]

$$
\begin{aligned}
P_{E\mathbf{X}}(R) := & \int_{\Omega} p_{E\mathbf{X}}(\mathbf{x},\mathbf{z})\mathbf{1}^{2}{}_{[(\mathbf{x},\mathbf{z})\in R]}\mathrm{d}(\mathbf{x},\mathbf{z}) = \\
& \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \int_{\Omega_{\mathbf{Z}}} p_E(\mathbf{z}|\mathbf{x})\mathbf{1}_{[(\mathbf{x},\mathbf{z})\in R]}\mathbf{dzdx} \qquad (2.38)\\
= & \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \left(\int_{\Omega_{\mathbf{Z}}} \delta(\mathbf{z}-E(\mathbf{x}))\mathbf{1}_{[(\mathbf{x},\mathbf{z})\in R]}\mathbf{dz} \right) \mathbf{dx} \qquad (2.39)\\
= & \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x})\mathbf{1}_{[(\mathbf{x},E(\mathbf{x}))\in R]}\mathbf{dx} \qquad (2.40)
\end{aligned}
$$

$$
\begin{aligned}
P_{G\mathbf{Z}}(R) := & \int_{\Omega} p_{G\mathbf{Z}}(\mathbf{x},\mathbf{z})\mathbf{1}_{[(\mathbf{x},\mathbf{z})\in R]}\mathrm{d}(\mathbf{x},\mathbf{z}) = \\
& \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \int_{\Omega_{\mathbf{X}}} p_G(\mathbf{x}|\mathbf{z})\mathbf{1}_{[(\mathbf{x},\mathbf{z})\in R]}\mathbf{dxdz} \qquad (2.41)\\
= & \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \left(\int_{\Omega_{\mathbf{X}}} \delta(\mathbf{x}-G(\mathbf{z}))\mathbf{1}_{[(\mathbf{x},\mathbf{z})\in R]}\mathbf{dx} \right) \mathbf{dz} \qquad (2.42)\\
= & \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z})\mathbf{1}_{[(G(\mathbf{z}),\mathbf{z})\in R]}\mathbf{dz} \qquad (2.43)
\end{aligned}
$$

Probability measures over regions on the image data $R_X \subseteq \Omega_X$ and the latent noise $R_Z \subseteq \Omega_Z$ can be similarly defined.

$$P_{\mathbf{X}}(R_{\mathbf{X}}) := \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x})\mathbf{1}_{[\mathbf{x}\in R_{\mathbf{X}}]}\mathbf{dx} \qquad (2.44)$$

$$P_{\mathbf{X}}(R_{\mathbf{X}}) := \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x})\mathbf{1}_{[\mathbf{x}\in R_{\mathbf{X}}]}\mathbf{dx} \qquad (2.45)$$

The optimal discriminator follows the same derivation from the GAN framework [24]. The Objective equation 2.35 can be reduced to the Jensen-Shannon divergence between joint distributions $P_{E\mathbf{X}}$ and $P_{G\mathbf{Z}}$ with the proof of the following propositions from the BiGAN [18].

Proposition 2.3.1. *For any E and G, the optimal discriminator $D^*_{EG} := \underset{D}{argmax} V(D,E,G)$ is the Radon-Nikodym derivative $f_{EG} := \frac{dP_{E\mathbf{X}}}{dP_{E\mathbf{X}}+dP_{G\mathbf{Z}}} : \Omega \mapsto [0,1]$ of measure $P_{E\mathbf{X}}$ with respect to measure $dP_{E\mathbf{X}} + dP_{G\mathbf{Z}}$.*

[2] Indicator function is a type of characteristic function that is used to indicate a membership on a subset X. It is denoted with bold $\mathbf{1}$ symbol.

Proposition 2.3.2. *The encoder and generator's objective for an optimal discriminator* $C(E,G) := \max_D V(D,E,G) = V(D^*_{EG},E,G)$ *can be rewritten in terms of the Jensen-Shannon divergence between the measures* P_{EX} *and* P_{GZ} *as* $C(E,G) = 2D_{JS}(P_{E\mathbf{X}} \parallel P_{G\mathbf{Z}}) - \log 4$

Discriminator is trained with both the encoder and generator network output. Therefore it is rational to assume that if we were to define a probability measure for discriminator . So we let P_{EG} be the average of $P_{E\mathbf{X}}$ and $P_{G\mathbf{Z}}$.

$$P_{EG} = \frac{P_{E\mathbf{X}} + P_{G\mathbf{Z}}}{2} \tag{2.46}$$

Since both $P_{E\mathbf{X}}$ and $P_{G\mathbf{Z}}$ are continuous with respect to the P_{EG}, the relationship of RN derivatives of f_{EG} and f_{GE} is defined accordingly:

$$f_{EG} := \frac{dP_{E\mathbf{X}}}{dP_{E\mathbf{X}} + dP_{G\mathbf{Z}}} \tag{2.47}$$

$$f_{GE} := \frac{dP_{G\mathbf{Z}}}{dP_{E\mathbf{X}} + dP_{G\mathbf{Z}}} \tag{2.48}$$

$$f_{EG} + f_{GE} := \frac{dP_{G\mathbf{Z}}}{dP_{E\mathbf{X}} + dP_{G\mathbf{Z}}} + \frac{dP_{E\mathbf{X}}}{dP_{E\mathbf{X}} + dP_{G\mathbf{Z}}} = \frac{d(P_{G\mathbf{Z}} + P_{E\mathbf{X}})}{d(P_{G\mathbf{Z}} + P_{E\mathbf{X}})} = 1 \tag{2.49}$$

Using the property of RN derivative (equation 2.24) we can transform the expected value function and rewrite the objective function of the BiGAN with a single probability measure.[18] [24]

$$\mathbb{E}_{\mathbf{x}\sim P}[g(\mathbf{x})] = \int_\Omega g \mathrm{d}P = \int_\Omega g \frac{\mathrm{d}P}{\mathrm{d}Q} \mathrm{d}Q = \int_\Omega g f_{PQ} \mathrm{d}Q = \mathbb{E}_{\mathbf{x}\sim Q}\left[f_{PQ}(\mathbf{x}) g(\mathbf{x})\right] \tag{2.50}$$

We start by rewriting the original objective function with the joint probability distributions.

$$V(D,E,G) = \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{E\mathbf{X}}}[\log D(\mathbf{x},\mathbf{z})] + \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{G\mathbf{Z}}}[\log(1 - D(\mathbf{x},\mathbf{z}))] \tag{2.51}$$

Then the transformation described in equation 2.50 is applied.

$$= \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{EG}}[\underbrace{2f_{EG}}_{\frac{\mathrm{d}P_{EX}}{\mathrm{d}P_{EG}}}(\mathbf{x},\mathbf{z}) \log D(\mathbf{x},\mathbf{z})] + \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{EG}}[\underbrace{2f_{GE}}_{\frac{\mathrm{d}P_{GZ}}{\mathrm{d}P_{EG}}}(\mathbf{x},\mathbf{z}) \log(1 - D(\mathbf{x},\mathbf{z}))] \tag{2.52}$$

The final stage of the equation falls into a type of function $f(a,y) = a\log y+(1-a)\log(1-y)$ [18] which provides $\underset{y}{argmax} f(a,y) = a$ for any $a \in [0,1]$. Thus the optimal Discriminator $D^*_{EG} = f_{EG}$. That proves the first proposition.

$$= 2\mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{EG}}\left[f_{EG}(\mathbf{x},\mathbf{z})\log D(\mathbf{x},\mathbf{z}) + f_{GE}(\mathbf{x},\mathbf{z})\log(1-D(\mathbf{x},\mathbf{z}))\right] \quad (2.53)$$

$$= 2\mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{EG}}\left[f_{EG}(\mathbf{x},\mathbf{z})\log D(\mathbf{x},\mathbf{z}) + (1-f_{EG}(\mathbf{x},\mathbf{z}))\log(1-D(\mathbf{x},\mathbf{z}))\right] \quad (2.54)$$

Second proposition can be proved with the same approach from GAN[24] using the connection acquired from the first proposition. [18]

First the objective equation is rewritten with the optimal discriminator and its equivalent RN derivative. [18]

$$C(E,G) = \max_D V(D,E,G) = V\left(D^*_{EG},E,G\right) \quad (2.55)$$

$$= \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{E\mathbf{X}}}\left[\log D^*_{EG}(\mathbf{x},\mathbf{z})\right] + \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{G\mathbf{Z}}}\left[\log\left(1-D^*_{EG}(\mathbf{x},\mathbf{z})\right)\right] \quad (2.56)$$

$$= \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{E\mathbf{X}}}\left[\log f_{EG}(\mathbf{x},\mathbf{z})\right] + \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{G\mathbf{Z}}}\left[\log f_{GE}(\mathbf{x},\mathbf{z})\right] \quad (2.57)$$

Then addition - subtraction of $\log 4$ is applied following the equation 2.21 to interpret the result as Jensen-Shannon divergence, hence proving the second proposition.

$$= \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{E\mathbf{X}}}\left[\log\left(2f_{EG}(\mathbf{x},\mathbf{z})\right)\right] + \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim P_{G\mathbf{Z}}}\left[\log\left(2f_{GE}(\mathbf{x},\mathbf{z})\right)\right] - \log 4 \quad (2.58)$$

$$= \mathrm{D}_{\mathrm{KL}}\left(P_{E\mathbf{X}}\|P_{EG}\right) + \mathrm{D}_{\mathrm{KL}}\left(P_{G\mathbf{Z}}\|P_{EG}\right) - \log 4 \quad (2.59)$$

$$= \mathrm{D}_{\mathrm{KL}}\left(P_{E\mathbf{X}}||\frac{P_{E\mathbf{X}}+P_{G\mathbf{Z}}}{2}\right) + \mathrm{D}_{\mathrm{KL}}\left(P_{G\mathbf{Z}}||\frac{P_{E\mathbf{X}}+P_{G\mathbf{Z}}}{2}\right) - \log 4 \quad (2.60)$$

$$= 2\mathrm{D}_{\mathrm{JS}}\left(P_{E\mathbf{X}}\|P_{G\mathbf{Z}}\right) - \log 4.\square \quad (2.61)$$

The proposition 2.3.2 allows us to characterize the objective function of the BiGAN in terms of the Jensen-Shannon divergence but implicitly it also proves just like in equation 2.26 that the global minimum of the training criterion is only achieved with the equality of the probability measures $P_{E\mathbf{X}}$ and $P_{G\mathbf{Z}}$. Consequentially this gives the optimal condition for both encoder

and generator.

Theorem 2.3.1. *if E and G are an optimal encoder and generator, then $E = G^{-1}$ almost everywhere; that is, $G(E(x)) = x$ for P_X almost every $x \in \Omega_x$, and $E(G(z)) = z$ for P_z, almost every $z \in \Omega_z$. [18]*

The inversion property of the BiGAN framework is dependent on the factor of $P_{E\mathbf{X}}$ being equal to $P_{G\mathbf{Z}}$ in the optimal state. Let R_X^0 be the region that the reconstructed image from the encoded noise is not equal to the input image and let R^0 be the region for the data, noise pairs which the data is also a member of the R_X^0. Using the optimal encoder and decoder, it can be proven that the probability measure for the region R_X^0 is zero, meaning that for almost every x in the region Ω_x, $G(E(x)) = x$ and vice versa.

$$R_X^0 := \{x \in \Omega_x : x \neq G(E(x))\} \tag{2.62}$$

$$R^0 := \{(x, z) \in \Omega : \mathbf{z} = E(x) \wedge x \in R_X^0\} \tag{2.63}$$

$$P_{\mathbf{X}}\left(R_{\mathbf{X}}^0\right) = \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \mathbf{1}_{\left[\mathbf{x} \in R_{\mathbf{x}}^0\right]} \mathbf{dx} \tag{2.64}$$

$$= \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \mathbf{1}_{[(\mathbf{x}, E(\mathbf{x})) \in R^0]} \mathbf{dx} \tag{2.65}$$

$$= P_{E\mathbf{X}}\left(R^0\right) \tag{2.66}$$

$$= P_{G\mathbf{Z}}\left(R^0\right) \tag{2.67}$$

$$= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \mathbf{1}_{[(G(\mathbf{z}), \mathbf{z}) \in R^0]} \mathbf{dz} \tag{2.68}$$

$$= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \mathbf{1}\left[\mathbf{z} = E(G(\mathbf{z})) \wedge G(\mathbf{z}) \in R_{\mathbf{X}}^0\right] \mathbf{dz} \tag{2.69}$$

$$= \int_{\Omega_{\mathbf{Z}}} \underbrace{p_{\mathbf{Z}}(\mathbf{z}) \quad \mathbf{1}_{[\mathbf{z}=E(G(\mathbf{z})) \wedge G(\mathbf{z}) \neq G(E(G(\mathbf{z})))]}}_{=0 \text{ for any } \mathbf{z}, \text{ as } \mathbf{z}=E(G(\mathbf{z})) \Longrightarrow G(\mathbf{z})=G(E(G(\mathbf{z})))} \quad \mathbf{dz} \tag{2.70}$$

$$= 0. \square \tag{2.71}$$

$$\tag{2.72}$$

Training of BiGAN framework is implemented using the same adversarial process of GANs. Encoder and generator network are trained together

since they both try to minimize the objective. Each iteration is divided into two parts. First the discriminator is trained using the outputs from the encoder and generator with fixed weights, pushing the gradient of the function towards a positive step. Also the parameters of discriminator θ_D is updated. Then θ_E and θ_G the parameters of encoder and generator are updated concurrently with a negative direction in the gradient of the objective function. [18]

Loss functions for the training optimization and Anomaly score computation differs in terms of the type of approach, different from AnoGAN[55]. AnoGAN framework used the combination of the reconstruction loss (equation 2.32) and the discrimination loss (equation 2.33) both to infer z from the query image and to compute the anomaly score. BiGAN framework's encoder is trained with the kind of function that is used to train the generator, sigmoid cross entropy. The advantage of the BiGAN is the reduced inference time to compute the anomaly score because the inferred noise from the query image $\hat{x}$ is the output from the generator $\hat{z} = E(\hat{x})$.

Anomaly score computation of BiGAN however, is the same approach used in AnoGAN. It uses a combination of the discriminaton loss and a reconstruction loss. The degree of norms used in the computation differs in the experiment stage to observe the behavior of the anomaly score. The Anomalous samples of the test dataset is determined by accepting the anomaly score that is greater then a predetermined threshold. This varies depending on the type of dataset and the dimensional complexity of the dataset the framework trained on. The implementation of this model contains the same generator and discriminator implementations as AnoGAN model. Details of the architecture can be found in the appendix section. (see A.2)

2.3.3 ALAD

Adversarially Learned Anomaly Detection, or ALAD [62] is a framework designed for the anomaly detection task. It builds upon the idea of AnoGAN, using the base architecture from the AliGAN and BiGAN with additional improvements incorporated to increase the stabilization of the adversarial training in both discriminator module and encoder, generator pair. I will first describe the improvements introduced to the adversarial training framework and discuss their potential advantages over the previous approach. Then I will explain the ALAD architecture in detail. Lastly, learning of the model and the anomaly score computation choices will be discussed.

2.3.3.1 Spectral Normalization

Even tough the training of GAN works in practice, it may easily become unstable in certain conditions. For example, in high dimensional space, density ratio estimation of the discriminator network is often inaccurate and this results in generator networks fail to learn the dimensional structure of the target data distribution. [6] Consequently, when the support of the model distribution and the target distribution is disjoint, then the discriminator may become decently efficient classifying real data from the generated data sample. In this scenario, the generator fails to learn to improve sample fake image data. Spectral Normalization is one of the recent proposed improvement methods to stabilize the weights of the discriminator network by bounding its gradients using Lipschitz continuity. [41]

Consider the optimal discriminator of $D(\boldsymbol{x},\theta) = \mathcal{A}(f(\boldsymbol{x},\theta))$ in equation 2.16. It takes the form :

$$D_G^*(\boldsymbol{x}) = \frac{q_{\text{ data}}(\boldsymbol{x})}{q_{\text{ data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})} = \text{sigmoid}\left(f^*(\boldsymbol{x})\right), \quad \text{where } f^*(\boldsymbol{x}) = \log q_{\text{ data}}(\boldsymbol{x}) - \log p_G(\boldsymbol{x}) \tag{2.73}$$

Where $\mathcal{A}$ is an activation function corresponding to the divergence of distance measure of the user's choice (usually a sigmoid). With the derivative of:

$$\nabla_{\boldsymbol{x}} f^*(x) = \frac{1}{q_{\text{data}}(\boldsymbol{x})} \nabla_{\boldsymbol{x}} q_{\text{data}}(\boldsymbol{x}) - \frac{1}{p_G(\boldsymbol{x})} \nabla_{\boldsymbol{x}} p_G(\boldsymbol{x}) \tag{2.74}$$

This gradient function can be unbounded or even incomputable depending on the distribution. Spectral normalization is applied to introduce regularity condition to derivative $f(\boldsymbol{x})$. [41]

Discriminator function can be modified to apply this regularization to the objective function:

$$\begin{aligned} &\arg\max V(G,D) \\ &\|f\|_{\text{Lip}} \leq K \end{aligned} \tag{2.75}$$

Where $\|f\|_{\text{Lip}}$ is depicted as the smallest value M such that:

$$\left\|f(\boldsymbol{x}) - f\left(\boldsymbol{x}'\right)\right\| / \left\|\boldsymbol{x} - \boldsymbol{x}'\right\| \leq M \tag{2.76}$$

for any x, x' with the norm being the l_2 norm. x_2 in X,[57]

Using the computation of the K-Lipschitz function of the each layer in discriminator, the spectral norm of each layer weight matrix can be normalized so the updated weights of the discriminator after each epoch of training becomes:

$$\overline{W}_{\text{SN}}(W) := W/\sigma(W) \tag{2.77}$$

Overall implementation of the spectral normalization addition to batch normalization proved useful for the performance of ALAD, particularly in the stabilization of the discriminator training. Model also employed this reparameterization in their architecture as well. As we will see in the chapter 5, addition of spectral normalization is also tested on our model.

2.3.3.2 Conditional Entropy (ALICE Framework)

The BiGAN framework in section 2.3.2 aims to learn both the mapping from the input image data to a latent representation with its encoder module and the inverse of that mapping relation with its generator module. It acquires these mappings by training its encoder and generator modules together in an adversarial manner to learn a joint distribution depicted in equation 2.37 and 2.36. The problem suggested by the ALICE framework[37] is that learning the joint distribution alone doesn't help the model to reconstruct images necessarily faithful to the reproductions of the input data. [37] This is because the objective function is built only to match joint distributions of image and noise pairs at the same time potential dependency structures or correlations between two random variables within each joint representation is not specified or constrained. [37]

In order to reduce identifiable issues associated with the objective of BiGAN, the conditionals $P_G(x|z)$ and $P_E(z|x)$ needs to be constrained and be under supervision using a paired samples from the domain Ω_x and Ω_z. Information-theoretic measure conditional entropy is used to provide this regularization.

Conditional entropy quantifies the amount of information needed to describe the outcome of a random variable Y given that the value of another random variable X is known [17] under the joint distribution

$$\pi(x, y), \quad x \in X, y \in Y$$

For the BiGAN framework the conditional entropies are defined below:

$$H^{\pi}(\boldsymbol{x}|\boldsymbol{z}) \triangleq -\mathbb{E}_{\pi(\boldsymbol{x},\boldsymbol{z})}[\log \pi(\boldsymbol{x}|\boldsymbol{z})] \tag{2.78}$$

$$H^{\pi}(\boldsymbol{z}|\boldsymbol{x}) \triangleq -\mathbb{E}_{\pi(\boldsymbol{x},\boldsymbol{z})}[\log \pi(\boldsymbol{z}|\boldsymbol{x})] \tag{2.79}$$

The Conditional entropy defined above is dependent on the underlying distribution of the random variables, in this case the distribution of the encoder and generator modules $p_G(x)$ and $p_E(z)$. In practice this value is intractable because we don't have access to the saddle points of the objective function beforehand. As an alternative, the approximation of the conditional entropy is calculated by bounding CE to the criterion of cycle-consistency [64]

Cycle consistency is the similarity of the reconstruction to the input data. Denoting the reconstruction of x as x', the cycle can be defined as:

$$x' = G(z'), \quad z' = E(x), \quad x \in \Omega_x, \quad x' \approx G(E(x)) \tag{2.80}$$

In other gan frameworks like Cycle-GAN[64], Disco-GAN [31] and Dual-GAN [61] cycle consistency constraint is implemented leveraging ℓ_k losses with $k = 1, 2$. Using ℓ_2 based pixel-wise loss functions to reconstruct the samples leads to blurry images during inference [34][37]. To enforce a better constraint and to improve the quality of the reconstructed samples, this framework suggests using the feature layer of the discriminator to compute the loss for the conditional entropy constraint. The objective function to approximate this constraint is given below:

$$\begin{aligned}\min_{\boldsymbol{E},\boldsymbol{G}} \max_{\boldsymbol{D}} \mathcal{L}_{\text{Cycle}}(E, G, D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \left[\log \sigma \left(f_{\boldsymbol{D}}(\boldsymbol{x}, \boldsymbol{x})\right)\right] \\ + \mathbb{E}_{\hat{\boldsymbol{x}}\sim p_{\boldsymbol{G}}(\hat{\boldsymbol{x}}|\boldsymbol{z}), \boldsymbol{z}\sim p_{\boldsymbol{E}}(\boldsymbol{z}|\boldsymbol{x})} \log \left(1 - \sigma \left(f_{\boldsymbol{D}}(\boldsymbol{x}, \hat{\boldsymbol{x}})\right)\right)\right]\end{aligned} \tag{2.81}$$

This additional adversarial training is implemented into the ALAD architecture to enforce a constraint to improve the quality of reconstructed noise and the image. Its addition is explained in the next part of the section.

2.3.3.3 ALAD Architecture

Training of the framework shares the same adversarial approach with BiGAN (2.3.2).Generator module simultaneously learns the mapping from input data to a latent representation while the encoder module learns the

opposite. Model also incorporates the mentioned improvements to increase the stabilization of the training procedure. The architecture overview is illustrated in figure 2.6

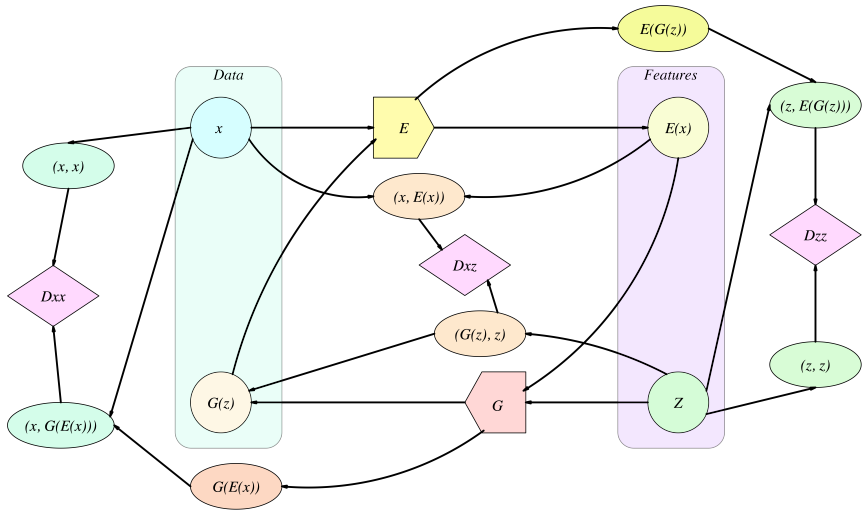


Figure 2.6 – ALAD architecture overview

Formally the ALAD architecture tries to match the joint distributions

$$p_E(x,z) = p_E(z|x)p_{\text{data}}(x)p_G(x,z) = p_G(x|z)p_Z(z) \tag{2.82}$$

and tries the separate the data, noise pairs $(x, E(x))$ and $(G(z), z)$ with an adversarial discriminator module D. For this framework we refer to the main discriminator as D_{xz}. The core objective function with the updated terms is :

$$\begin{aligned} V(D_{xz}, E, G) = &\mathbb{E}_{x\sim p_{\text{data}}}[\log D_{xz}(x, E(x))] \\ &+ \mathbb{E}_{z\sim p_Z}[1 - \log D_{xz}(G(z), z)] \end{aligned} \tag{2.83}$$

Section 2.3.3.2 explains the convergence problem of the joint distribution. Because of the lack of constraints imposed on the p_{data} and p_Z, the objective function may not converge properly and this consequentially decreases the quality of the reconstructed samples, namely $x' \approx G(E(x))$. In addition to the image reconstruction, to enforce a constraint on the noise encoding from the input image, $z' \approx (E(G(z)))$, ALAD architecture also adds a second objective function to the overall adversarial training loop to regularize the conditional distributions. The CE term for the image and for

the noise is defined below.

$$H^{\pi}(x|z) = -\mathbb{G}_{\pi(x,z)}[\log \pi(x|z)] \tag{2.84}$$

$$H^{\pi}(z|x) = -\mathbb{E}_{\pi(x,z)}[\log \pi(z|x)] \tag{2.85}$$

As explained in the previous section, since the saddle points beforehand can't be obtained for the computation, these terms can be approximated with the discriminators D_{xx} and D_{zz} respectively. The training of these discriminators is added to the main adversarial training loop as a part of the discriminator module training. Their objective functions are defined in equation 2.86

$$\begin{aligned} V(D_{xx}, E, G) = {} & \mathbb{E}_{x \sim p_{\text{data}}}[\log D_{xx}(x, x)] \\ & + \mathbb{E}_{x \sim p_{\text{data}}}[1 - \log D_{xx}(x, G(E(x)))] \end{aligned} \tag{2.86}$$

$$\begin{aligned} V(D_{zz}, E, G) = {} & \mathbb{E}_{z \sim p_{\mathcal{Z}}}[\log D_{zz}(z, z)] \\ & + \mathbb{E}_{z \sim p_{\mathcal{Z}}}[1 - \log D_{zz}(z, E(G(z)))] \end{aligned} \tag{2.87}$$

Putting it all together, ALAD framework solves the following combination of value function during the adversarial training.

$$\begin{aligned} & \min_{G,E} \max_{D_{xz}, D_{xx}, D_{zz}} V(D_{xz}, D_{xx}, D_{zz}, E, G) = \\ & V(D_{xz}, E, G) + V(D_{xx}, E, G) + V(D_{zz}, E, G) \end{aligned} \tag{2.88}$$

During the training, the generator and the encoder are updated simultaneously. In the second part of the adversarial training, all the discriminators update their weights to both regularize the reconstructions and to converge to equilibrium for the main value function. Spectral normalization is used in every convolutional layer of the discriminators. Framework also implemented them to the layers of the encoder network with the purpose of further increasing the quality of the reconstructed samples. [62].The result of the training and its analysis will be discussed in chapter 5.

The Anomaly score computation of the framework is based on the quality of the reconstructed query samples. If the reconstructed query image is distant to the original input it will obtain a greater anomaly score. ALAD framework suggests that because of the nature of the spatial data, even though there are similarities, obtained noise during reconstruction may mask potential feature similarities, preventing their detection. The sug-

gested approach of the framework is to compute the anomaly score using the feature layer of the D_{xx} discriminator shown in equation 2.89.

$$A(x) = \|f_{xx}(x,x) - f_{xx}(x, G(E(x)))\|_1 \tag{2.89}$$

In this equation, $f(\cdot,\cdot)$ represents the vector of activations of the layer before the output of the D_{xx} discriminator [62]. Anomaly score $A(\cdot)$ captures the confidence such that the given image is well encoded and reconstructed by the generator and therefore it is from the real data distribution. [62] However different kinds of anomaly score computations are also considered for the sake of comparison.

These are (including the one discussed):

$$\begin{aligned} \bullet L_1 &: A(x) = \left\|x - x'\right\|_1 \\ \bullet L_2 &: A(x) = \left\|x - x'\right\|_2 \\ \bullet \text{ Logits } &: A(x) = \log\left(D_{xx}\left(x, x'\right)\right) \\ \bullet \text{ Features } &: A(x) = \left\|f_{xx}(x,x) - f_{xx}\left(x, x'\right)\right\|_1 \end{aligned} \tag{2.90}$$

Overall this framework aims to improve the generator and encoder learning with improvements such as spectral normalization and conditional entropy discriminators. Its performance with our dataset and ways to improve the adversarial training will be discussed in chapter 5.

2.3.4 GANomaly & Skip-GANomaly

GANomaly framework [2] and Skip-GANomaly [3] are the most recent models that focuses on the anomaly detection using adversarial training. Their performance on the benchmark datasets like CIFAR-10 [33] and SVHN [43] exceeds the previous mentioned frameworks. This section will explain their structure and their differences in terms of one another.

GANomaly and Skip-Ganomaly models differ with a significant change in the model itself. Both models contain a generator and discriminator modules but generator module is actually composed of an adversarial autoencoder network. Details of the both architecture will be discussed in order.

Generator module of the GANomaly framework composed of an encoder-decoder network as can be seen in figure 2.7. The third module is the

encoder network for the generated samples. Generator network takes the image data as an input and sequentially it creates the latent representation of the image and reconstruction from the latent representation then the second encoder network encodes the latent representation of the reconstructed sample. Discriminator network shares the same functionality as in GAN framework. [24]

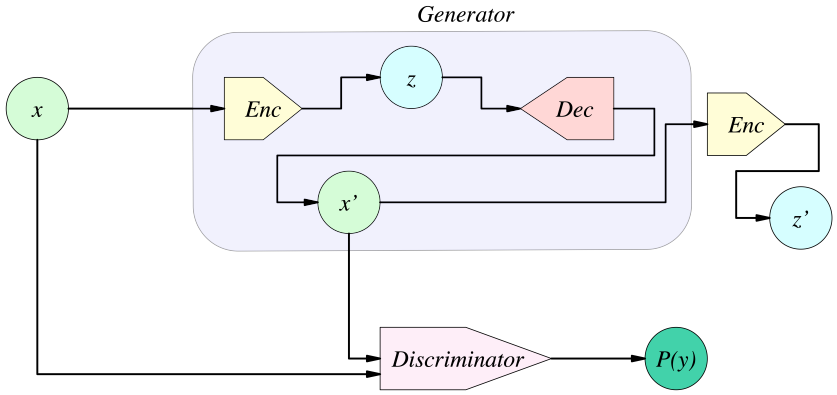


Figure 2.7 – GANomaly architecture overview

With the use of the adversarial autoencoders, framework obtains a superior reconstruction ability compared to the previous models. But in doing so, its generative sub-module decoder is not a true generative component from the perspective of the generative adversarial networks. With the change of the model architecture its training objective also changes. GANomaly framework uses adversarial training scheme but its generator module loss is the combination of 3 separate loss functions. These are:

- Adversarial Loss
- Contextual Loss
- Encoder Loss

Adversarial loss represents the loss function component for the adversarial training of the framework. Instead of maximizing the probability of discriminator's attaining generated image as real, it has a loss function derived from the [53] that uses feature matching. The main purpose of this type of loss function is to reduce the instability in the GAN training. Adversarial loss depicted in equation 2.91 is the $\mathcal{L}_2$ distance of the feature representation of the original and the generated image, respectively. [2] Feature layer is extracted from the activation layer before the output of the

discriminator.

$$\mathcal{L}_{adv} = \mathbb{E}_{x \sim p_{\mathbf{x}}} \| f(x) - \mathbb{E}_{x \sim p_{\mathbf{x}}} f\left(G(x)\right) \|_2 \tag{2.91}$$

Only adversarial loss does not help to optimize the generator towards reconstructing images similar to the input data. To this end, contextual loss is also added to the generator loss function. According to [27], using $\mathcal{L}_1$ instead of $\mathcal{L}_2$ decreases the blurriness in the generated images, so the following contextual loss that measures the distance between original image and the generated one is added to the to the total generator loss.

$$\mathcal{L}_{con} = \mathbb{E}_{x \sim p_{\mathbf{X}}} \| x - G(x) \|_1 \tag{2.92}$$

Additional encoder loss is also added to the generator to improve the encoding of the latent representation of the reconstruction. This loss function aims to minimize the distance between the encoded latent representation of the input image and the reconstruction. The Loss function is depicted below.

$$\mathcal{L}_{enc} = \mathbb{E}_{x \sim p_{\mathbf{X}}} \| G_E(x) - E(G(x)) \|_2 \tag{2.93}$$

Overall, the objective loss function of the generator is defined in equation 2.94 where the weight parameters w_{adv}, w_{con} and w_{enc} are used to change the impact of individual losses.

$$\mathcal{L} = w_{adv}\mathcal{L}_{adv} + w_{con}\mathcal{L}_{con} + w_{enc}\mathcal{L}_{enc} \tag{2.94}$$

GANomaly framework uses encoder loss function also as an anomaly score measure.

$$\mathcal{A}(\hat{x}) = \| G_E(\hat{x}) - E(G(\hat{x})) \|_1 \tag{2.95}$$

Skip-GANomaly framework[3] is the continuation of the GANomaly architecture with changes mainly to the generator network. As can be seen from figure 2.8, encoder and decoder networks inside the generator framework is connected using skip connections.

At each layer of the encoder, convolutional feature map is concatenated to the corresponding decoder layer. Addition of skip connections provides a better reconstruction capability to the generator than the GANomaly framework. [2]. Generator objective function is built using the same logic. Because of the exclusion of the secondary encoder network, the encoder loss

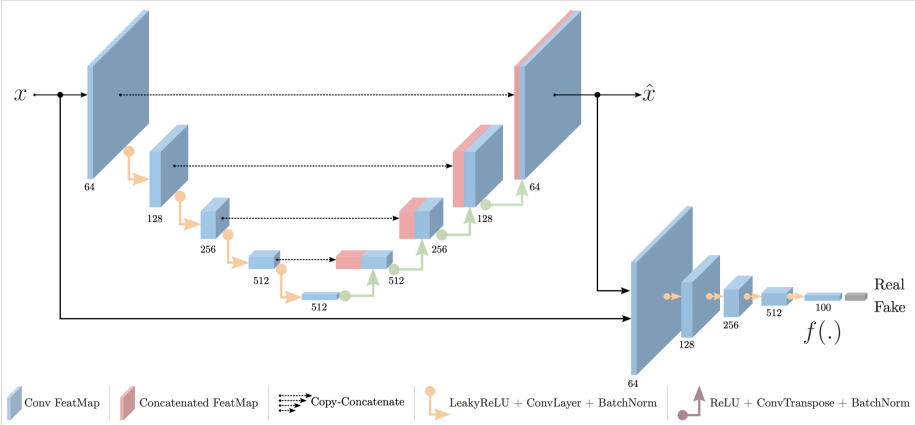


Figure 2.8 – Skip GANomaly architecture overview [3]

is obtained using the feature layer of the discriminator.

$$\mathcal{L}_{enc} = \mathop{\mathbb{E}}_{x \sim p_x} |f(x) - f(\hat{x})|_2 \tag{2.96}$$

To find anomalies during the inference stage, Skip-GANomaly framework combines the encoder loss with the use of contextual loss.

$$\mathcal{A}(\dot{x}) = \lambda R(\dot{x}) + (1 - \lambda) L(\dot{x}) \tag{2.97}$$

$R(\dot{x})$ represents the contextual loss with the reconstruction term and $L(\dot{x})$ represents the encoder loss with the latent representation term. Term λ is used to adjust the impact of the loss functions to the overall anomaly score.

Both GANomaly and Skip-GANomaly uses adversarial encoder architecture in their generator networks and therefore have superior reconstruction capabilities compared to the previous 3 framework. The primal problem with their architecture is that they don't adapt the true distribution of the input data. They learn the underlying latent representation well to reconstruct the input and detect anomalies. However their generator network represents the scenario where generator has learnt the input data distribution fairly well. In order to improve the results of a GAN based anomaly detection framework, insights gained from these 2 networks are significantly important. Their contribution to the improved framework and their performance analysis regards to other models will be explored in chapters 4 and 5 respectively.

CHAPTER 3

Latest Developments

This chapter presents the developments in the field of anomaly detection and generative adversarial networks. Although the models mentioned in the state of the art chapter deliver certain level of performance, potential improvements are still a possibility. These improvements are related to both adversarial training of the generator and discriminator networks and different strategy to stabilize the training of encoder network present in the overall framework. In the following sections, some of these improvements that are later adopted by our framework will be introduced and their contribution to solve the disadvantages we observed in the previous frameworks will be discussed.

3.1 F-AnoGAN Framework

AnoGAN [55] is considered as the first framework that uses generative adversarial networks for an anomaly detection task. The main problems with this framework, are the stabilization issues of the adversarial training, and the second stage of the framework which was mapping from the latent representation to the input data distribution which can be also considered as the inverse mapping of the generator. The model used back propagation to approximate the latent representation for every query image to compute the anomaly score which resulted a very poor performance in terms of the computation time. This was the main disadvantage of the framework because it is very challenging to integrate into a real life application with an implausible inference time. F-AnoGAN (Fast AnoGAN) framework [54] aims to eliminate this issue by implementing a new training strategy. It

also uses a new objective function for the adversarial training to further stabilize the generator discriminator performance. In the rest of this section, F-AnoGAN framework, its training strategies and anomaly detection methods will be discussed.

The framework architecture of F-AnoGAN is very similar to the BiGAN [18] framework. It consists of a generator discriminator for the adversarial learning and an encoder network to learn the inverse mapping from latent representation to the input image data.

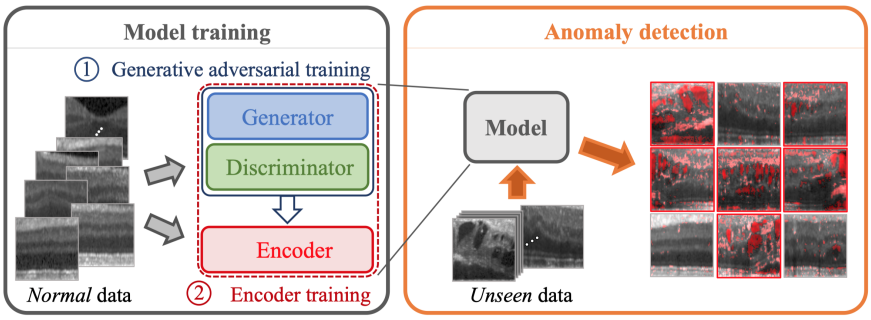


Figure 3.1 – F-AnoGAN Framework Overview [54]

The first improvement is the change of the training style of the whole framework. In BiGAN approach, encoder and generator is trained simultaneously to fool the discriminator. Hence discriminator is modified to classify the pairs of noise (latent representation) and images instead of the single image approach used in the GAN framework. Therefore it tries to distinguish samples from a joint distribution which explained in section 2.3.2. Including encoder to the adversarial setup also introduces instability issues which ALAD Framework [62] tried to mitigate with additional discriminators that approaximate conditional entropy. (section 2.3.3.2). F-AnoGAN framework addresses this issue by separating the training of the encoder from the generator discriminator pair. The new framework can be seen in figure 3.2.

This new training scheme comprises of two stages. In the first stage GAN framework is trained. The objective function for the adversarial training is also replaced with the Wasserstein GAN's objective function [7] although training of the GAN can be performed with any other framework's approach (including the original GAN framework) according to the [54].

In the second stage, encoder network is trained using generator and discriminator with locked weight. Two separate pipelines are used to measure

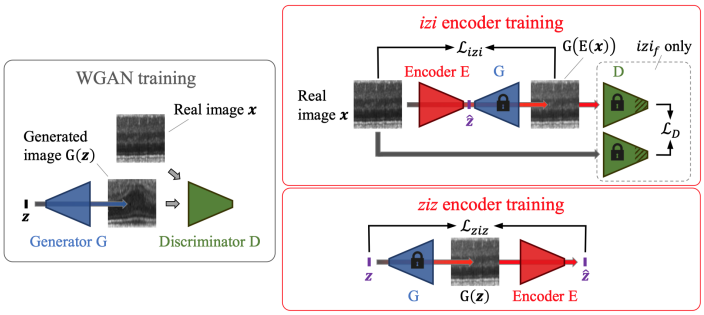


Figure 3.2 – F-AnoGAN Training Strategies [54]

the performance of the encoder. These are IZI (image-noise-image) and ZIZ (noise-image-noise) respectively.

IZI Encoder Training

IZI method follows the standard autoencoder network approach. The generator network with fixed weights acts as a decoder network. During the training, same image dataset used in the first stage for training GAN is mapped to a latent mapping z by a trainable encoder and then reconstructed using the fixed generator network. Training objective for this setup is to minimize MSE (Mean Squared Error) based reconstruction error of the input image x and the reconstructed image $G(E(x))$.

$$\mathcal{L}_{izi}(\mathbf{x}) = \frac{1}{n}\|\mathbf{x} - G(E(\mathbf{x}))\|^2 \tag{3.1}$$

This approach has an important drawback regarding to latent representation. Since the distribution of the latent representation of the input image is not known, the encoder is trained only using a form of contextual loss which enforce the similarity only in the image space. Without the sufficient information about the latent space, encoder may map images to a representations such that the reconstructions are not convincing enough for the discriminator to evaluate as real. [54]. Therefore the framework suggests also including a latent space based loss function derived from the feature layer of the discriminator to guide the encoder training. Improved IZI_f training objective is defined below.

$$\mathcal{L}_{izi_f}(\mathbf{x}) = \frac{1}{n} \cdot \|\mathbf{x} - G(E(\mathbf{x}))\|^2 + \frac{\kappa}{n_d} \cdot \|f(\mathbf{x}) - f(G(E(\mathbf{x})))\|^2 \tag{3.2}$$

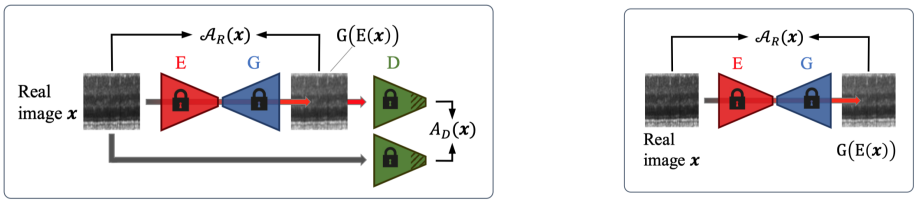
where the $f(\cdot)$ represents the feature layer of the discriminator as an additional statistics, n_d is the dimensionality of the intermediate feature representation f and κ denotes the weighting factor for the inclusion of the discriminator guidance.

ZIZ Encoder Training

Different from the IZI method, this approach reverses the auto encoder setup and forms a decoder encoder architecture. A random sampled noise from the z-space is mapped to the image space using the fixed generator and then generated sample is encoded using the trainable encoder network. The loss for the training is defined as the MSE based reconstruction of the noise which depicted in equation below.

$$\mathcal{L}_{ziz}(\mathbf{z}) = \frac{1}{d}\|\mathbf{z} - E(G(\mathbf{z}))\|^2 \tag{3.3}$$

The shortcoming of this approach is that even though the encoder is trained with a loss that will enforce a similarity in latent space, encoder sees only images that are generated by the generator. It doesn't see any image samples from the training dataset which affects the contextual similarity of the reconstruction of learned latent space distribution.



(1) Anomaly Score computation scheme for izi and izi_f method

(2) Anomaly Score computation scheme for ziz architecture

Figure 3.3 – Anomaly Score computations for both training methods [54]

To calculate the anomaly score for inference, deviation of the query images from their reconstructions are quantified. Figure 3.3 represents the anomaly score computations for both training methods.

IZI_f method uses the same loss functions for the computation of the anomaly score. Anomaly score for image x is defined as:

$$\mathcal{A}(\mathbf{x}) = \mathcal{A}_R(\mathbf{x}) + \kappa \cdot \mathcal{A}_D(\mathbf{x}) \tag{3.4}$$

$$\mathcal{A}_R(\mathbf{x}) = \frac{1}{n} \cdot \|\mathbf{x} - G(E(\mathbf{x}))\|^2 \tag{3.5}$$

$$\mathcal{A}_D(\mathbf{x}) = \frac{1}{n_d} \cdot \|f(\mathbf{x}) - f(G(E(\mathbf{x})))\|^2 \tag{3.6}$$

For IZI and ZIZ method, the computation of the anomaly score reduces down to $\mathcal{A}_R(\mathbf{x})$ function only. Both training methods yields a similar anomaly score computation for the anomalous samples. Since the models are trained with a dataset that doesn't contain any anomalies, the query images that have anomalous regions results in poorer reconstruction hence a higher anomaly score while the normal images produce a more similar reconstructions to the original sample.

The Significance of this framework is that it separates the training of the encoder from the adversarial setting of the GAN's while preserving the inverse mapping functionality for the inference while keeping the inference time of the model relatively short. Chapter 4 will explain its contribution to the proposed improved framework.

3.2 Energy Based Generative Adversarial Networks

Training a GAN framework perhaps the most sensitive part of working with them. There has been an extensive amount of research to improve the stabilization of the adversarial training or perhaps offer a new methodology. [53] and [6] offer number modifications to the original adversarial training setup to improve the generation performance ,increase the stabilization and prevent issues such as mode collapse[1]. [7] and [25] offers a new method to compute the loss and ways to stabilize it. Mainly because of its adversarial setting, training of GAN frameworks will always be a topic of interest. Energy based generative adversarial networks [63] offers a new training method by redefining the loss functions and the functionality of the discriminator network. This section will introduce the architecture and how it works.

[1]Mode collapse is the scenerio where generator learns enough information to fool the discriminator very early. It finds a generation that fools the discriminator and does not learn anything else from the discriminator. Because of that all the generated samples from the generator look the same, defeating the purpose of generating from the distribution of the data.[6]

Energy based models measures the current state of the model by assigning a scalar predefined energy as a degree of compatibility [35]. Training energy based models consists of mapping low energy outputs to the samples with the "correct" class, and mapping higher energy outputs to the "incorrect" class. From the perspective of GANs, EBGAN model is trained to ouput lower energy for the real data, while the generated data (fake images) outputs a higher energy. The model architecture can be seen in figure 3.4.

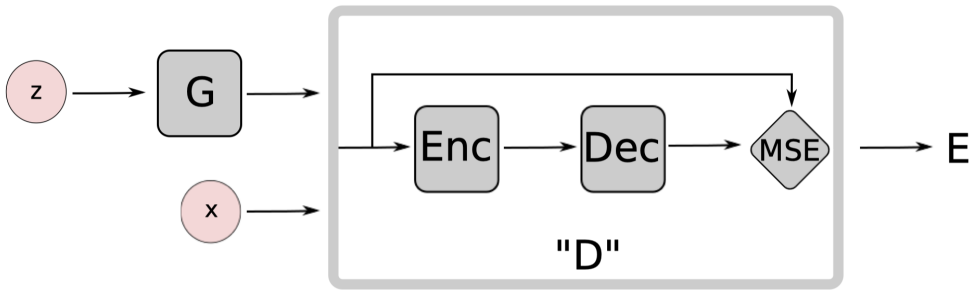


Figure 3.4 – EBGAN Model Structure [63]

The main structural difference of this framework from the other ones is that discriminator is defined as an auto encoder network to provide the reconstruction of the inputs. The energy value function is defined as the margin loss of the input and the reconstructed sample [63] as a reconstruction error.

Formally a margin is used to create an energy gap between the correct output and the incorrect output. For data sample x, and a generated sample $G(z)$ with z being sampled from a known distribution p_z, The discriminator loss $\mathcal{L}_D$ and the generator loss $\mathcal{L}_G$ are formally defined in equation 3.7.

$$\begin{aligned} D(x) &= \| \operatorname{Dec}(Enc(x)) - x \| \\ \mathcal{L}_D(x,z) &= D(x) + [m - D(G(z))]^+ \\ \mathcal{L}_G(z) &= D(G(z)) \end{aligned} \tag{3.7}$$

where $D(x)$ is the reconstuction loss and $[\cdot]^+ = \max(0, \cdot)$. Minimizing $\mathcal{L}_G$ with respect to the generator is similar to maximizing the second term of the $\mathcal{L}_D$. Instead of calculating the probability of the input being real or fake, discriminator is responsible for attaining the correct amount of energy to the input images.

Using autoencoder network as a discriminator brings additional advantages to the training setting. Rather than using a single target information to train the model, reconstruction based output offers a more diverse targets for the discriminator [63]. One of the reasons that causes stabilization issues in the adversarial training is that discriminator might provide gradients not useful enough for generator to improve the quality of the generated samples. Having a reconstruction loss output might provide a better gradient in terms of different directions as opposed to binary logistic loss[63]. Another advantage is that even though they are trained with an adversarial setting, autoencoder networks might prove useful to learn the underlying energy manifold of the data without supervision or negative examples when they are trained using a proper regularization constraint. This means that in this setting generated samples and input data are both used to model the underlying energy manifold of the data.

EBGAN framework argues that, the discriminator network is regularized by exposing generated samples from the generator which it should output higher reconstruction energies [63]. The benefit of this regularization is that instead of a hand crafted predefined one, the regularization term can also be trained along with the discriminator. Adversarial training allows a direct connection between learning the energy function and producing contradictive samples in that regard.

EBGAN framework also adds additional penalty called pulling away term (PT) that run at a representation level to prevent mode collapse depicted in equation 3.8.

$$f_{PT}(S) = \frac{1}{N(N-1)} \sum_{i} \sum_{j \neq i} \left(\frac{S_i^\top S_j}{\|S_i\| \, \|S_j\|} \right)^2 \tag{3.8}$$

S is the feature output from the encoder for generated images. Pulling-away term measures the cosine similarity among all generated images features S in a minibatch. If the mode collapses, the feature vectors will thereupon be similar, i.e. the angles are close to zero and the cosine will max-out. Therefore, it will add a high penalty if there are too similar. [63]

EBGAN framework offers a more flexible approach to train the discriminator and generator network. Transition from convolutional based discriminator to autoencoder network also provides a potential inclusion for the computation of the anomaly score. Its impact on the proposed framework will be discussed in the next chapter.

CHAPTER 4

Architectural Improvements

This chapter presents the modifications applied to aforementioned approaches to improve the performance of the anomaly detection. All the discussed models measures its performance metric on well known datasets, such as CIFAR-10 [33] and SVHN [43]. The dataset used in this thesis presents additional challenges to this problem we aim to solve. The first section will introduce its dataset to the reader and will give examples. Next section will discuss the shortcomings of the previous approaches regarding the interpretation of the dataset and detection of the existent anomalies. Sections 4.3 and 4.4 will explain the modified architecture and the significance of the changes.

4.1 SEM Image Dataset

Nanofibrous materials acquired increasingly significant demand from variety of fields in the Industry. It constitutes a foundation material for a lot of products including areas in medicine, filtration, sensors and manufacturing applications. [13]. Despite the demand and continuous research development towards its production, manufacturing nanofibrous materials is still challenge for scaled of mass production. Several techniques for producing nanofibers have been presented in the literature. [13]. Electro spinning method is the focus of this anomaly detection task. It produces a structure which consists of filaments woven in with randomized geometric pattern. You can see one of the images of the material produced without any anomalies in figure 4.1.

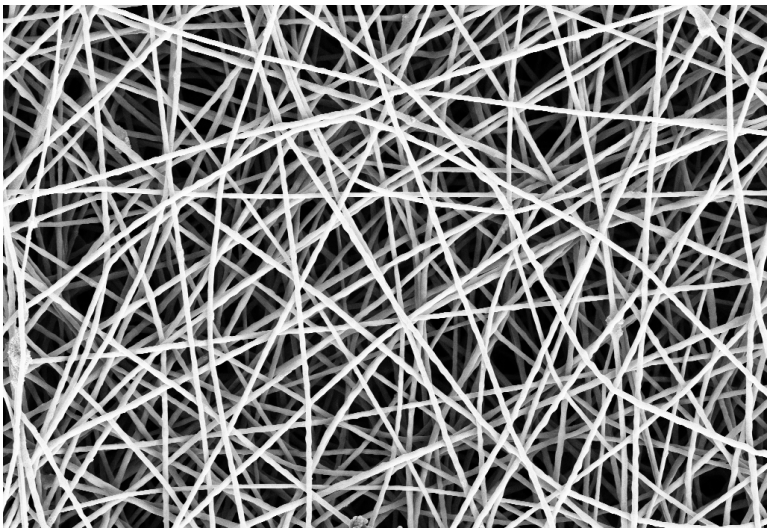
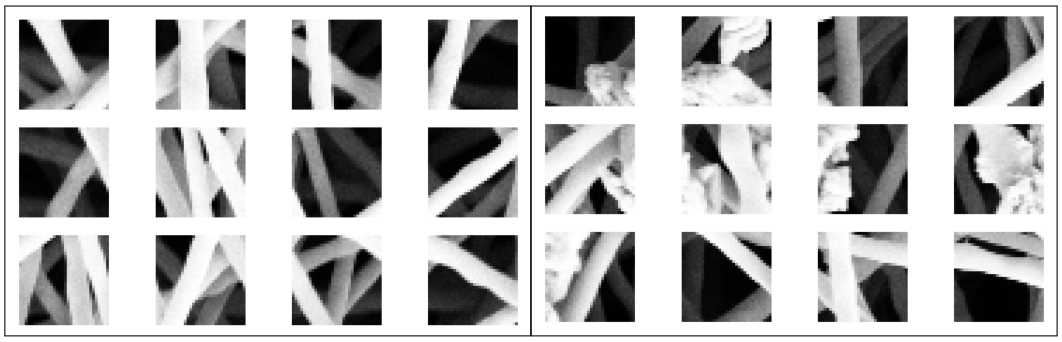


Figure 4.1 – Part of Training Dataset from the SEM Image Dataset [56]

Dataset consists of 5 images with no anomalies and 40 sample images that have various types of anomalous regions. To preserve computational efficiency without sacrifising from performance, training dataset and testing dataset is sampled from SEM image dataset.



(1) Normal regions **(2)** Anomalous regions

Figure 4.2 – Normal and Anomalous region patches for the training and testing

32×32 patches are selected as the image size as if proposed framework prove useful for the anomaly detection, testing the model with other known datasets such as CIFAR10 [33] and SVHN [43] would be more con-

venient. 28×28 patch size is also considered but choosing 28×28 image size forced model to have less transposed convolution layers (see chapter A for model details) and GAN models designed with that architecture experienced model collapse frequently in the preliminary experiments. Figure 4.21 shows the example patches used for the training phase. Training dataset consists of patches which contains no anomalies. Figure 4.22 shows the anomalous samples which are used in the inference stage. Anomalous regions can reside in both the topmost layer or can be hidden in deep in the woven structure.

4.2 Analysis of Aforementioned Approaches

This section presents an analysis on the performance of GAN based state of the art anomaly detection methods presented in section 2.3 and a discussion to identify the shortcomings of these methods. Stabilization issues in the adversarial training stage, reconstruction problems related to the encoding of latent representation and the method of computing the anomaly score will be discussed. While discussing these issues, the proposed model will be introduced incrementally by addressing the modifications and measures to mitigate the affect of these shortcomings to the performance.

Proposed methods in chapter 2 can be further divided into 2 separate categories with respect to their generator structure.

- Pure GAN (AnoGAN (2.3.1), BiGAN (2.3.2) and ALAD (2.3.3))
- Autoencoder Variants (GANomaly and Skip-GANomaly (2.3.4))

First group has a generator that accepts the noise as an input and uses adversarial training to match the generated sample distribution to the input data distribution. The latter uses an autoencoder based decoder in favor of the generator but still employs adversarial training to learn the latent representation. During the analysis of the first two observations, autoencoder variants will not be considered for discussion since their generator network does not suffer from the stabilization issues of GANs.

4.2.1 Stabilization of Adversarial Training

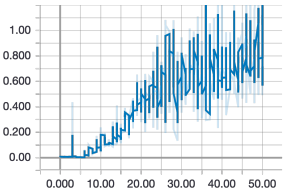
Generator and discriminator's objective function stabilization is still an important issue in GAN training. Various approaches and modifications are experimented to further improve the training of the GANs and prevent non convergent scenerios. [6] and [53] proposed additional stabilization "tricks" to improve the convergence properties of the objective function and prevent

mode collapse. These include adding noise with a decay over time to both input image and generated sample before putting through discriminator to add a factor of robustness to the discriminator, using soft labels to define the true and fake member class instead of the binary truth values and flipping labels of the true and generated images to fool the discriminator even more and provide higher gradient flow to generator early on in the training to help it learn to generate images better [53].

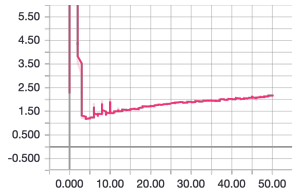
In the training phase of an generative adversarial network, loss values of generator does not portray a traditional convergence line in plots. Adversarial mini max game prevents the losses of the player networks to converge to a certain limit. If the loss is reaching zero in any case, it indicates a problem in the learning capacity of one of the networks. If discriminator loss converges to zero too quickly, it means that it learned to discriminate between the real images and the ones generated by the generator network. On the other hand if the generator loss converges to zero early in the training it indicates a mode collapse but situation may arise in even normal looking training sessions. The problem is at some point in the training generator generates an image that perfectly fools the discriminator. If gradients obtained from this loss computation is high enough, generator might stop learning from the gradient flow of discriminator and start to generate the same sample in each iteration. This sample also may not be visually similar to the target distribution. This eliminates the purpose of having a generator network.

To mitigate these shortcomings, ablation study is performed on all Pure GAN models which consists of previously mentioned training improvements with same model capacity and training hyperparameters to create an equal environment for the models. Despite models proving a certain performance standard on the benchmark datasets ([33, 43]), they performed poorly on the SEM image dataset ([56]). More importantly, the training of the networks were unstable enough to prevent to obtain a consistent performance benchmark. Obtaining a stable GAN for generating reconstructions is fairly important because it is the first step of the anomaly detection framework. Figure 4.3 presents the generator/discriminator learning graphs and their generation samples from the final training epoch.

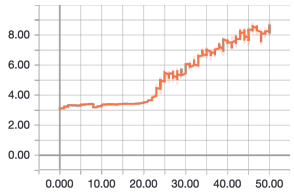
As seen in the figure, discriminator is gradually learning to differentiate between the real and generated image while the generator network does not seem to set its loss to a certain level as desired. This trend is observed in the majority of the experiments and in the ablation study as well. From the perspective of the image generation, samples generated from GANs are not



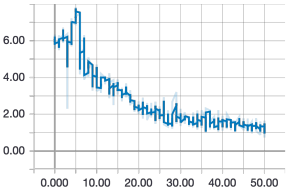
(1) AnoGAN generator training



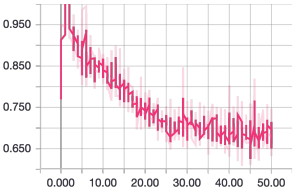
(2) BiGAN generator training



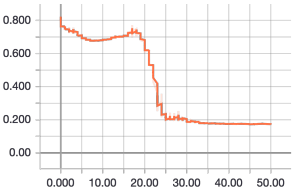
(3) ALAD generator training



(4) AnoGAN discriminator training



(5) BiGAN discriminator training



(6) ALAD discriminator training

(7) AnoGAN generated image sample

(8) BiGAN generated image sample

(9) ALAD generated image sample

Figure 4.3 – Training information for the Pure GAN models. For graphs, x axis is the number of epochs and the y axis is the loss value

contextually sound enough to represent the target distribution of the SEM image dataset. To obtain a more concise observation the reconstructions of these GANs will also be inspected.

$$\begin{aligned} D(x) &= \|\,\mathrm{Dec}(Enc(x)) - x\| \\ V(G,D) &= D(G(z)) + D(x) + [m - D(G(z))]^+ \\ \mathcal{L}_D(x,z) &= D(x) + [m - D(G(z))]^+ \\ \mathcal{L}_G(z) &= D(G(z)) \end{aligned} \tag{4.1}$$

Considering the results of these experiments, training objective of the EBGAN

model is adopted to obtain a more stable training with our dataset. (See eqn. 4.1)

Defining loss function as a value of energy still preserves the adversarial aspect of the objective function. An auto encoder network acts as a discriminator module and attains an energies to the real and fake image samples with high energy values assigned to generated images. Generator tries to "fool" the discriminator to decrease the amount of energy fake image is assigned. Here the energy term is described as the ability of discriminator to reconsturct the given output, so it is a reconstruction error based value.

Using energy based GAN proved useful to obtain a consistent performance without having issues with argued stabilization problems. Details regards to its training and model's reconstruction capability will be discussed in its own section.

4.2.2 Convergence of Encoder Training

Stabilizing the GAN training is actually the second step for a reconstruction based anomaly detection approach. As proposed, generator network learns the mapping from the latent space Z to the image space X. However in order to test unknown images for anomalies, the latent representation of the image must be extracted. From this point on this operation is called the "inverse mapping". Generators are not able to provide this kind of inverse mapping so pure gan based models followed different approaches to obtain this information.

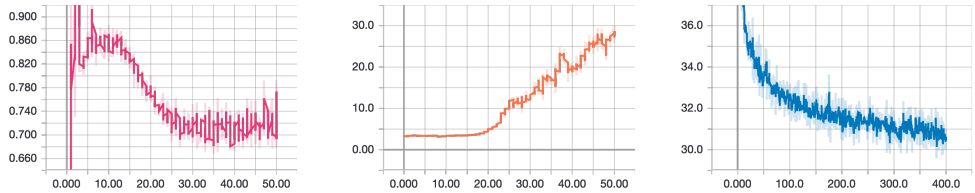
AnoGAN model implements a back propagation routine for a predetermined number of steps to approximate the latent representation of the query image. It randomly initializes a latent representation vector and updates it until the loss that is defined as the ℓ_2 norm of the query image and generated sample converges. The main disadvantage of this approach is the time constraint. Method works with image patches so processing whole image takes a considerable amount of time compared to other methods.

BiGAN and ALAD on the other hand, adds an additional encoder network to their adversarial training to learn the inverse mapping while learning the image distribution (see eqn. 4.2).

$$\begin{aligned} V(D,E,G) = &\mathbb{E}_{x\sim p_{\text{data}}}\left[\log D_{xz}(x,E(x))\right] \\ &+ \mathbb{E}_{z\sim p_Z}\left[1-\log D_{xz}(G(z),z)\right] \end{aligned} \tag{4.2}$$

To learn both mappings, models learn the joint probability distribu-

tion of the image and the noise pairs. Discriminator network therefore is modified to process this new representation (see implementation details in appendix A.2 and A.3). Integrating an encoder network solves the time constraint problem because the latent representation is acquired with feeding the query image to the encoder network at the inference stage and the computation time is negligible compared to AnoGAN's approximation. Nonetheless addition of a new component to the adversarial training brings new stabilization problems with regards to encoder network. Training of the generator discriminator, implicitly affects the quality of the representation encoded. ALAD implements additional discriminators to control the conditional probability distributions of the learned joint distribution from both ends (For noise and for image) to improve the encoder performance but experiments did not provide a considerable advance even with ablation study. Figure 4.4 presents the training graphs of the models and their obtained reconstructions. AnoGAN model does not have a training graph for encoder since it uses a back propagation and it would have a convergence graph for each patch in the dataset. Instead, the training graph of the first iteration of the proposed model's encoder is presented.



(1) BiGAN Encoder Training

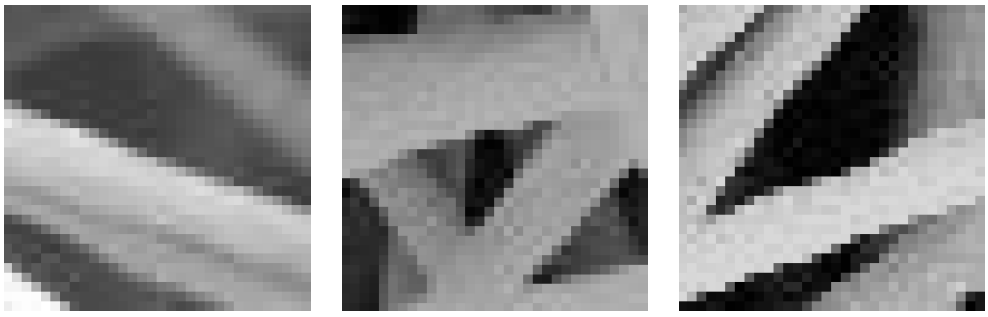
(2) ALAD Encoder Training

(3) ENCEBGAN Encoder Training

Figure 4.4 – Training graphs of pure GAN models that implement encoder network and the first iteration of the proposed network

By itself, these training graphs does not give us a reliable insight about encoder performance. Other than its quantitative results, encoder networks performance can be quantified with models' ability to reconstruct a given image after training. At this point in pipeline, impact of a good generator also comes into play. In both models, the training of the encoder network is computed by feeding the latent representation obtained from the encoder and feeding it to the generator to reconstruct the training data. So performance and proper training of the encoder network also depends on the adversarial training of generator. In the mini max game played by BiGAN and ALAD, both encoder and generator tries to fool the discriminator. Therefore stabilization problem of generator inadvertently affects

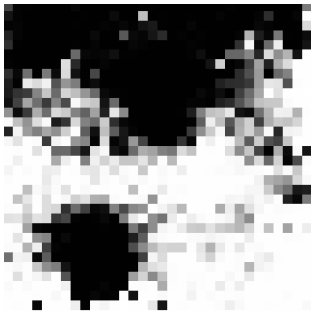
the encoder convergence which also affects the quality of the reconstructions obtained from the model. Some distortions and the information loss is expected in the generator based reconstruction. In our dataset, these imperfections cause reconstructed samples to be interpreted as anomalous sample though the input is a normal sample. Figure 4.5 shows the reconstructions from AnoGAN, BiGAN and ALAD's inference stage.



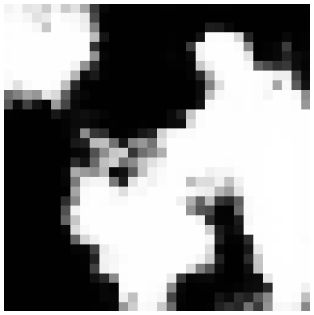
(1) AnoGAN Query Image

(2) BiGAN Query Image

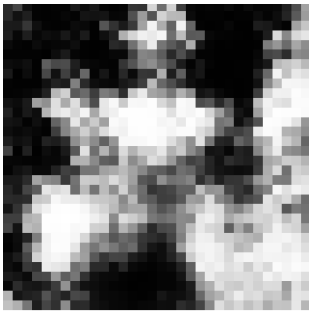
(3) ALAD Query Image



(4) AnoGAN Reconstruction



(5) BiGAN Reconstruction



(6) ALAD Reconstruction

Figure 4.5 – Reconstructed samples from the pure GAN models. Upper row is the given query image and the bottom row is their reconstructions

Proposed model in this work also uses additional encoder network to get the inverse mapping information. Several modifications are applied to the training in order to improve the stabilization and ensure the convergence of the encoder. Eqn. 4.1 propose the modified generator discriminator approach. Addition of the encoder network and its training is also based

on an energy based training.

$$
\begin{aligned}
D(x) &= \|\operatorname{Dec}(Enc(x)) - x\| \\
V(G,E,D) &= D(G(z)) + D(x) + [m - D(G(z))]^{+} + D(G(E(x))) \\
\mathcal{L}_E(x) &= D(G(E(x)))
\end{aligned}
\tag{4.3}
$$

Like AnoGAN and ALAD, proposed model also trains encoder network with generator networks help to obtain the reconstruction. However, instead of training all the networks in the adversarial setting, proposed model separates the training of the generator discriminator pair and the encoder network. Training strategy presented in F-AnoGAN framework (see section 3.1) proposes an initial adversarial training. In the second part of the training the encoder network is trained by creating a pipeline with the other network components with fixed weights (inference mode) (See figure 3.2). This approach focuses on the encoder training objective by removing the dependency between encoder and generator which in turn also decreases the potential stabilization issues of the generator network.

So far, configuring the models' objective function to an energy based setting and separating the joint training into sequential steps give the model two opportunities:

- The stabilization of the GAN training, which means better reconstruction performance.
- Easy convergence for encoder network because it isn't trained concurrently with the generator so learning latent reprensetation is uni directional gradient descent.

Next section will discussed the model with the mentioned changes and continue to analyze the existent shortcomings despite the improvements.

4.3 Encoded Energy Based Generative Adversarial Network

The first iteration of the proposed model is called encoded energy based generative adversarial network,or ENCEBGAN. This section presents how its built, its training scheme and the issues it solved.

ENCEBGAN consists of an energy based generative adversarial network framework with an additional encoder network. The roles of the networks are identical to BiGAN (2.3.2). Two main differences are the change in the

rationale in adversarial training component and separation of the encoder network training. Figure 4.6 presents the network components with an emphasis on the training sequence.

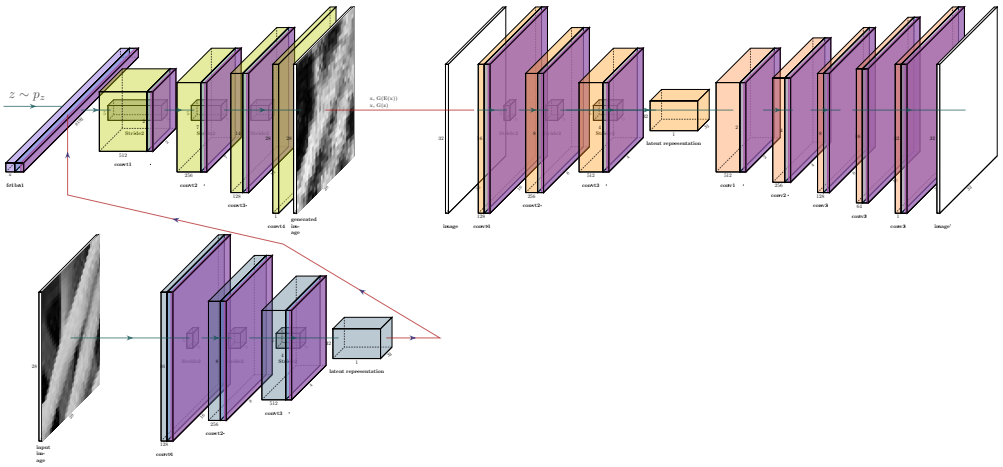


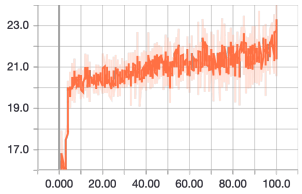
Figure 4.6 – ENCEBGAN Model Overview

Training of the model is divided into two sections. In the first section generator and discriminator trains adversarially to learn the mapping from image space to latent space. While generator learns the mapping, discriminator is concurrently learning to reconstruct the training dataset and learns to encode the training data on its own. This latent representation will be used later on as an auxiliary information for anomaly score computation. After the adversarial training is completed, pipeline is created from the encoder generator networks to form a conceptual autoencoder and whole network is trained using the reconstruction error based objective function for the encoder network. In second stage, generator and discriminator don't learn any additional information about the dataset. This approach proved to be useful to stabilize both generator and encoder hence increased the reconstruction quality of our model. Anomaly score computation is the same as the pure gan based methods which is the reconstruction error between the query and the output of encoder generator pipeline as depicted in equation 4.4.

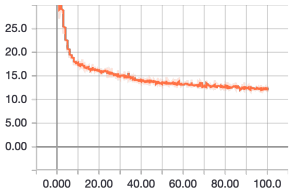
$$\mathcal{A}(x) = \|x - G(E(x))\|^2 \tag{4.4}$$

Figure 4.7 presents the training graph information and reconstructed samples as a comparison to reconstructions in figure 4.5.

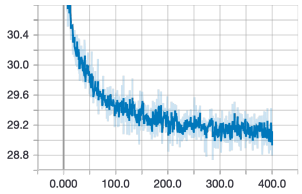
With previously mentioned architectural improvements, ENCEBGAN performed better than BiGAN and ALAD and obtained the same perfor-



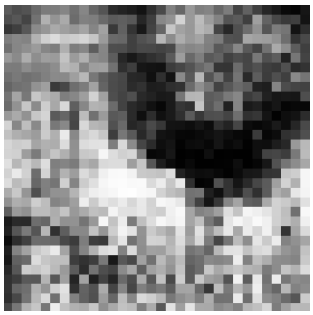
(1) Generator Training Graph



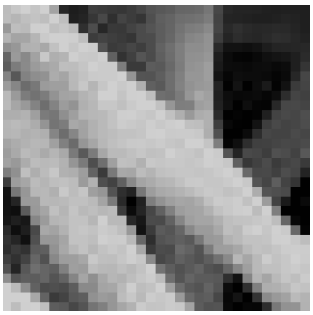
(2) Discriminator Training Graph



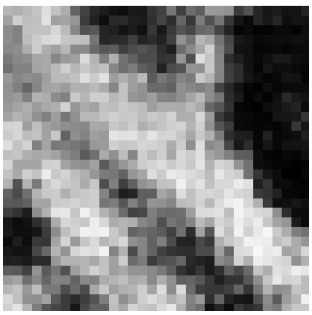
(3) Encoder Training Graph



(4) ENCEBGAN Generated Sample



(5) ENCEBGAN Query Sample



(6) ENCEBGAN Reconstruction

Figure 4.7 – Training graphs and qualitative examples from ENCEBGAN Model

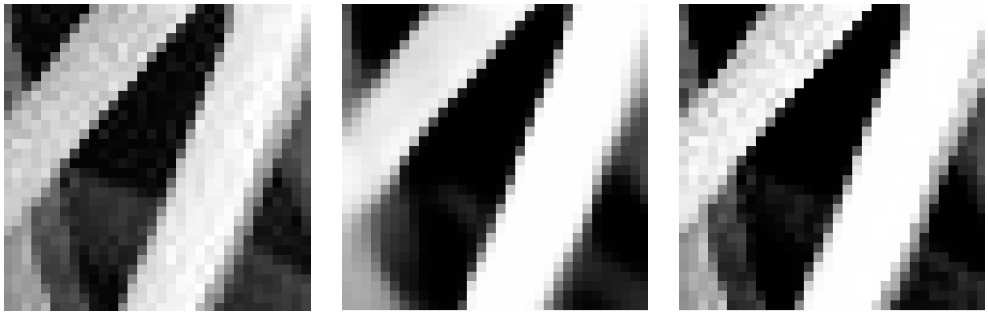
mance as AnoGAN which will discussed in chapter 5. Separating the training of encoder and changing the adversarial objective function mechanics consequently stabilized the convergence of both networks and the reconstruction quality has increased compared to the pure GAN based models.

Imperfections in the reconstructions are usually expected in reconstruction based anomaly detection methods since the network trained to learn the latent representation is regularized to prevent overfitting. In SEM image dataset, anomalous regions can occur in any level of the image with varying levels of brightness. During the experiments of the models it is observed that, imperfections in the reconstructions may be identified as anomalous regions even though the image is anomaly free. This issue inadvertently cripples the anomaly detection performance of ENCEBGAN (and other pure gan based models).

Next section will discuss the anomaly score computation methods featured in both pure GAN based and autoencoder variant models, and explan the addition of a secondary encoder network to improve the overall performance of the proposed model.

4.3.1 Anomaly Score Computation

Previously we divided the state of the art GAB based anomaly detection methods into two main categories. Pure GAN based models and autoencoder variants. The main difference between these groups is that decoder (generator equivalent) network of these models is trained only with the latent representation of the image supplied from the encoder network. Since they learn to reconstruct using the encoded version of the training data, their reconstructions are superior compared to the pure GAN based methods. Figure 4.8 shows an example reconstruction from both GANomaly and Skip-GANomaly 2.3.4.



(1) Query Sample

(2) GANomaly Reconstruction

(3) Skip-GANomaly Reconstruction

Figure 4.8 – Reconstruction examples of Autoencoder Variant Models

Architectural difference between GANomaly and Skip-GANomaly is the additional skip connections between the encoder and decoder network of the latter. This information transfer enables Skip-GANomaly to retain more information about the data and help it to accumulate this data across the layers during the decoding stage of the reconstruction. One interesting observation is that even though the general quality of the Skip-GANomaly model is better than GANomaly on the SEM image dataset, ablation study performed in chapter 5(see Table 5.5 and 5.6) shows that overall GANomaly prediction performance is superior compared to Skip-GANomaly.

Both anomaly score computation is based on the reconstruction based error, but GANomaly uses a secondary encoder network to learn the latent representation of the reconstruction and uses this information to compute the anomaly score (see eqn. 2.95). Skip-GANomaly on the other hand composes the contextual data (reconstruction loss) with the latent loss which is obtained from the feature layer of its discriminator network (see eqn. 2.97). We speculate that training a secondary encoder network improved

GANomaly model's interpretation of the anomalous sample data. Erroneous predictions that are caused due to misinterpreting imperfect reconstructions as samples that contains anomalous regions may be averted by interpreting both input image and its reconstruction at the latent dimensional complexity.

Considering these observations and related improvements, next section will present the final version of the proposed model for anomaly detection. General architecture of the model, its training procedure and its inference stage for anomaly detection will be discussed.

4.4 Sequentially Encoded Energy Based Generative Adversarial Network

Sequentially encoded energy based generative adversarial network is the main proposed model that aims to mitigate the disadvantageous outcomes of stabilization and convergence problems experienced vy the previously mentioned methods (see chapter 2). Even though it is presented incrementally throughout the chapter, its general architecture, training strategy and its approach to anomaly detection problem will be discussed.

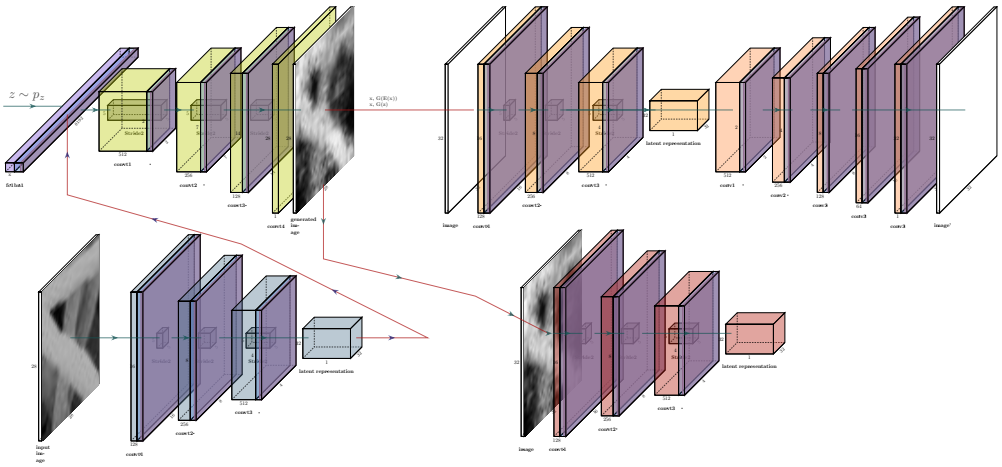


Figure 4.9 – SENCEBGAN Model Overview

SENCEBGAN model consists of an energy based generative adversarial network and 2 additional encoder networks. The first encoder network learns the representation of training data and its called generator encoder or E_G. Second encoder learns the latent representation of the reconstructions

of the training data provided by the generator network. This encoder network will be mentioned as reconstruction encoder or E_R. Training objective of the model is given in equation 4.5.

$$\begin{aligned} D(x) &= \|\operatorname{Dec}(Enc(x)) - x\| \\ V(G, E_G, E_R, D) &= \underbrace{D(G(z))}_{\text{Generator}} + \underbrace{D(x) + [m - D(G(z))]^+}_{\text{Discriminator}} + \\ &\quad \underbrace{D(G(E_G(x)))}_{\text{Encoder}_G} + \underbrace{\|E_G(x) - E_R(G(E_G(x)))\|^2}_{\text{Encoder}_R} \end{aligned} \tag{4.5}$$

Training objective is divided into three separate sequences. In the first stage of the training, energy based gan framework is trained. After the training is completed, autoencoder architecture is formed by merging E_G and Generator G with fixed weights. Combined with discriminator (also fixed weights) E_G is trained using the reconstruction error produced from the temporary autoencoder configuration. In the last stage, E_R is trained to learn the latent representation of the reconstructed samples. Training is performed using the ℓ_2 norm of the encoded representation residual (see eqn. 4.6).

$$\begin{aligned} \mathcal{L}_{E_G}(x) &= D(G(E(x))) \\ \mathcal{L}_{E_R}(x) &= \|E_G(x) - E_R(G(E_G))(x)\|^2 \end{aligned} \tag{4.6}$$

SENCEBGAN model separates anomaly detection computation into image based and latent dimension based categories. Different anomaly score computations are tested to measure the impact of the additional encoder networks and observe both implicit encoding and reconstruction capability of the discriminator network.

$$\begin{aligned} \mathcal{A}_R(x) &= \|x - G(E_G(x))\|^2 \\ \mathcal{A}_{R_D}(x) &= \|D(x) - D(G(E_G(x)))\|^2 \\ \mathcal{A}_{IC}(x) &= \lambda \cdot \mathcal{A}_R(x) + (1 - \lambda) \cdot \mathcal{A}_{R_D} \end{aligned} \tag{4.7}$$

Score functions defined in equation 4.7 defines the anomaly measure using spatial data. $\mathcal{A}_R(x)$ calculates the reconstruction error during the inference stage. This anomaly score is the one used in all pure GAN based models. In order to test the reconstruction capability of the discriminator $\mathcal{A}_{R_D}(x)$ score is employed. This score is calculated by performing 2 consecutive reconstructions on the sample image. $\mathcal{A}_{IC}(x)$ is image based com-

bination anomaly score implemented to measure each of the image based anomaly scores' affect on an ensemble approach.

$$\begin{aligned} \mathcal{A}_L(x) &= \|E_G(x) - E_R(G(E_G(x)))\|^2 \\ \mathcal{A}_{L_D}(x) &= \left\|E_G(x) - L_{D_Z}(G(E_G(x)))\right\|^2 \\ \mathcal{A}_{L_{D_Z}}(x) &= \left\|L_{D_Z}(x) - L_{D_Z}(G(E_G(x)))\right\|^2 \\ \mathcal{A}_{LC}(x) &= \lambda \cdot \mathcal{A}_L + (1-\lambda) \cdot \mathcal{A}_{L_{D_Z}} \end{aligned} \tag{4.8}$$

Secondary encoder network E_R added to the model's pipeline enabled model to capability to inspect the normality of query images in latent dimensional space and provided an alternative medium regards to anomaly score computation. Inspired by the GANomaly model's (see sec. 2.3.4) evaluation method, various types of anomaly score evaluations based on the latent representation space is depicted in equation 4.8. $\mathcal{A}_L(x)$ measures anomaly score based on the ℓ_2 norm of the difference between encodings obtained during the pipeline. Encoding capability of the discriminator network is also explored in these category. $\mathcal{A}_{L_D}(x)$ and $\mathcal{A}_{L_{D_Z}}(x)$ scores tests the anomaly score by comparing the latent representation obtained by the discriminator network. Finally, combined anomaly score for latent representation is employed to measure the affect of each score to an ensemble methods.

In terms of the performance, proposed model performed better than all previously mentioned models except GANomaly. Performance analysis and observations regarding to its potential improvements and different configurations will be explored in chapter 5.

CHAPTER 5

Experimental Results

This chapter presents the performance of all the models introduced in this thesis including the state of the art gan based anomaly detection methods (see section 2.3) and the proposed model SENCEBGAN which is introduced on chapter 4. All experiments were run on a server containing Titan X. In particular, we will investigate the AUROC scores of the models with their corresponding precision and recall capacities across different cases.

Organization of this chapter as follows: Section 5.1 gives a detailed explanation of the experimental setting providing obtaining the training and testing data and configurations of the models. Section 5.2 introduces the performance metrics that are used to evaluate and compare the models. Following two sections, section 5.3 and 5.4 describe the experiments conducted on the state of the art models and proposed model. Finally, section 5.5 discusses the results of the experiments.

5.1 Experiment Settings

Training and testing data for the experiments are obtained from the SEM image dataset [56]. Dataset contains 5 normal image samples with the resolution of 1024×696 and 40 Anomalous images samples (same resolution) with a corresponding mask image that denotes where each anomalous region is located. Training dataset is created by cropping 32×32 patches randomly from the normal images. Test dataset is obtained using the same method but with an overlapping sequential approach instead of randomization. Smaller patch size (28×28) is also tested for preliminary

gan experiments. Due to its size, designed generator network has one less combined transposed convolution layer (Transposed Convolution + Batch Normalization + Leaky ReLu) and some of the models we introduced in chapter 2 were designed for minimum 32×32 images. Therefore 32×32 image patches are used throughout the experiments.

For all experiments, same hyper parameter values regarding the optimizer used in the training are used for all models. In terms of training, batch size and number of epochs are also fixed to equally measure the learning capacity of the generator discriminator networks contained in the models. Other related hyper parameter selections will be mentioned in their respective sections throughout this chapter.

5.2 Performance Metrics

This section will introduce the performance metrics used for the interpretation of the experiments performed. These are:

- Precision
- Recall
- F1 Score
- AUROC (Area Under Receiver Operating Characteristic curve)

To define these performance metrics, first the statistical measures used in classification problems will be introduced. Suppose we have a classification problem with a certain dataset. In this problem we have a condition or a feature we want to identify and in general some part of the dataset has this condition and the remaining part does not. If the prediction regarding to this condition is correct for the sample this is called **true positive**.**Sensitivity** measures the true positive rate in a classification problem. Same as the positive, **true negative** is correct rejection of the condition. **Specifity** is therefore described as the proportion of the true negative samples that are correctly rejected by the classification. There are 2 types of errors defined in this context. **Type 1** error refers to the rate of falsely identifying a negative sample as positive. It is also called the **false positive rate**. **Type 2** error measures the error rate of false rejections of the positive samples which is also defined as the **false negative rate**. Our performance measures are composed of these primal statistical measures.

Recall is the ability of a model to find all the relevant cases within a dataset. In our case, detection of all the anomalous examples contained in the test set would indicate a perfect recall. Its score range is $[0, 1]$ inclu-

sive. Increasing recall capacity decreases the false negative rate to correctly identify all possible target class samples. **Precision** on the other hand is the ability of a classification model to identify **only** the relevant data points. Increased precision means a decrease in the false positive rate because a system favors precision aim towards eliminating all false predictions. The calculation of the both metrics is given below

$$\textbf{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{5.1}$$

$$\textbf{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{5.2}$$

Precision and recall comprises a trade off situation. If the model favors the precision, the recall capacity decreases because eliminating the false positives inadvertently increases the false negative rate and vice versa. To give equal importance to both metrics, **F1 score** is used. The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$\mathrm{F}_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}$$

The main metric we use to interpret the model performance is area under receiver operating characteristic curve, **AUROC** for short. ROC curve visualizes the trade of relationship between the recall and precision capacity as the threshold for identifying a positive in the model changes. The threshold is the indicator value above which a sample is considered in the positive class. In our problem, threshold is determined from the anomaly score. ROC curve plots the true positive rate on the y-axis versus the false positive rate on the x-axis. (See figure 5.1) True positive rate is actually recall. False positive rate is the probability of false detection for the system. Calculations for these rates are provided below:

$$\textbf{True Positive Rate} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{5.4}$$

$$\textbf{False Positive Rate} = \frac{\text{false positives}}{\text{true negatives} + \text{false positives}} \tag{5.5}$$

the AUROC value can be obtained by calculating the area under the ROC curve which has a range between 0 and 1 with a higher number indi-

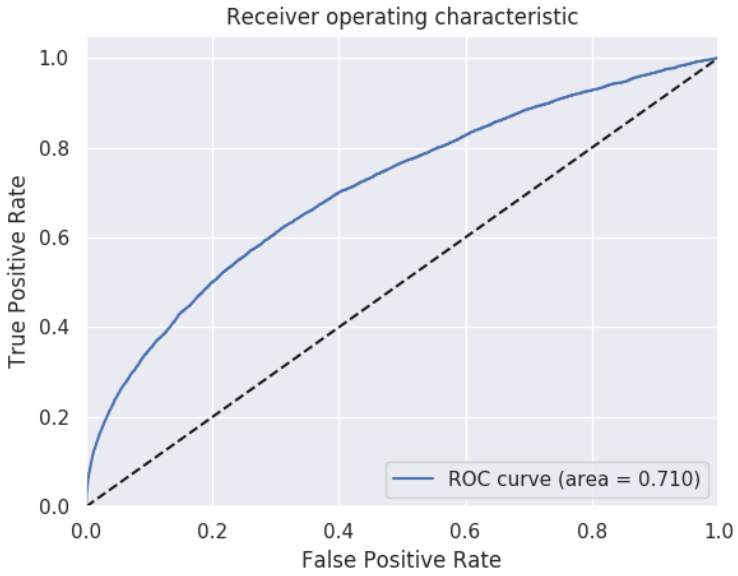


Figure 5.1 – ROC Curve Example with AUROC value

cating better classification performance.

In the performed experiments, AUROC value is considered as the main comparison metric amongst the models. Recall and precision are used to give an auxiliary information about the model's prediction accuracy and capacity. As a reporting criterion, highest F1 score obtained from the percentile interval between 80% and 99.9% is as a threshold is used to determine the optimal precision and recall values.

5.3 State of the Art GAN Based Model Experiments

This section is devoted to the performance analysis of pure GAN based models. In the initial experiment stage, all state of the art GAN based models are implemented using the same generator and discriminator architecture. Since all the models are tested on known benchmark datasets and there is no experiment with our dataset, all models' initial run is acquired as a baseline. Baseline performances are reported in the table 5.1.

Table 5.1
Baseline performance values of the state of the art GAN based models

Models	**Metrics**			
	AUROC	Precision	Recall	F1 Score
AnoGAN	0.59941	0.14253	0.27783	0.18841
BiGAN	0.62699	0.10201	0.32227	0.15497
ALAD	0.44842	0.05521	0.17443	0.08388
GANomaly	0.77697	0.39701	0.31356	0.35038
Skip-GANomaly	0.53182	0.07379	0.23314	0.11211

After this initial test, it is observed that pure GAN based models performed poorly on our target dataset and only GANomaly model obtained a full precision/recall curve even though it is below 0.5. Precision recall curve is an alternative way of visualizing the precision and recall capacity of a model. The definition of precision states that if the number of predictions are equal to the number of anomalies contained in the dataset then the precision is exactly 1.0. In order for that to be happen, model should not make a prediction unless it is absolutely certain hence with this criterion in mind the model has an greater false negative rate. So if the precision is 1.0 that means with a given threshold the model (acceptance point such that if sample has a greater or equal score than threshold it is considered as anomaly) can always make the right prediction. From the perspective of recall, it also means that there is a threshold value that enables model to find all anomalies contained in the dataset even if it increases the overall false positive rate of prediction performance.

In the ablation studies, mainly 3 heuristic improvements are tested to improve the stabilization of the training and consequently improve the performance. These are:

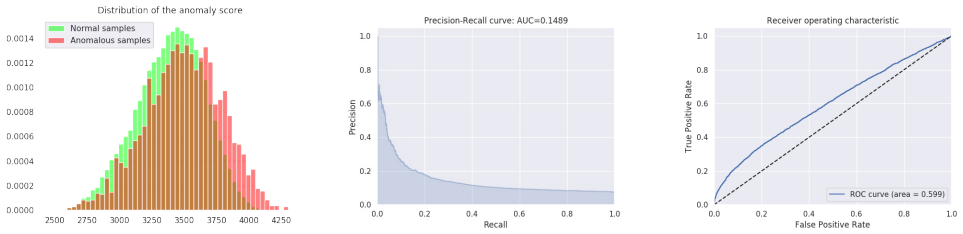
- (IN) Inclusion of noise to the real input images and generated images before feeding them into discriminator network to confuse discriminator.
- (SL) Using soft labels for the class representation instead of hard 0,1 labeling for the cross entropy computation used in the adversarial loss
- (LF) Flipping labels of the samples that are fed to the discriminator. Since the discriminator knows the type of image provided to it, mixing the labels helps to confuse discriminator so that gradient flow to the generator does not decrease.

In the ablation study of AnoGAN (see table 5.2), overall performance

Table 5.2
Ablation study for AnoGAN to test the effect of various training improvements for stabilization.

Model		Metrics			
		AUROC	Precision	Recall	F1 Score
AnoGAN	Normal	0.59941	**0.14253**	0.27783	0.18841
	IN	0.59978	0.14010	0.27309	0.18512
	SL	0.43092	0.06875	0.13401	0.09087
	LF	0.43319	0.06822	0.17732	0.09854
	IN + SL	**0.59987**	0.12877	**0.33468**	**0.18958**
	IN + LF	0.52329	0.09075	0.23587	0.13107
	SL + LF	0.53469	0.09440	0.24534	0.13634
	LF + SL + IN	0.54625	0.09973	0.25922	0.14405

couldn't pass the 0.6 mark. Addition of the noise and using soft labels improved the overall score by a small margin but decreased the precision capacity of the model. Recall capacity is slightly improved with the addition of noise and soft labelling. Using label flipping and soft labelling alone did not contribute any improvement on the contrary it pulled down the AUROC value below 0.5. Separation histogram, precision/recall trade-off and ROC curve of the best configuration is provided in figure 5.2.



(1) Separation Histogram

(2) Precision/Recall Trade off

(3) ROC Curve

Figure 5.2 – Visual performance analysis info for the best configuration of ablation study of AnoGAN

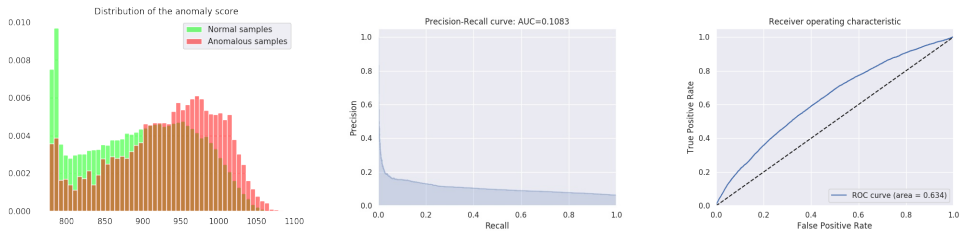
Table 5.3 shows the ablation study for the BiGAN model. We observe that baseline performance is only marginally improved by the addition of noise and label flipping. Single heuristic additions considerably made worse the overall performance with the exception of using soft class labels. Another aspect about the general performance is that even though its AU-

ROC performance is higher than AnoGAN, precision of the model achieved a lower score while improving its total recall value.

Table 5.3
Ablation study for BiGAN to test the effect of various training improvements for stabilization.

Model		Metrics			
		AUROC	Precision	Recall	F1 Score
BiGAN	Normal	0.62699	0.10201	0.32227	0.15497
	IN	0.38195	0.03721	0.11756	0.05653
	SL	0.58904	0.08251	0.26066	0.12534
	LF	0.39541	0.03532	0.11160	0.05366
	IN + SL	0.52646	0.06978	0.22045	0.10600
	IN + LF	**0.63362**	**0.10917**	**0.34490**	**0.16585**
	SL + LF	0.57855	0.08483	0.26800	0.12887
	LF + SL + IN	0.62912	0.10617	0.33542	0.16129

Considering the whole capacity of the model, we observe in figure 5.3 that the highest capacity and recall values along the threshold axis decreased considerably.



(1) Separation Histogram

(2) Precision/Recall Trade off

(3) ROC Curve

Figure 5.3 – Visual performance analysis info for the best configuration of ablation study of BiGAN

Figure 5.4 shows the reconstructions of given input images during the training. Convergence issue mentioned in section 4.2.2 supports the results provided here as can be seen from the training progression, there is no visual resemblance between the input images and their corresponding reconstructions.

Ablation study for ALAD model is presented in table 5.4. Unlike AnoGAN and BiGAN, ALAD couldn't pass the 0.5 mark performance on its own.

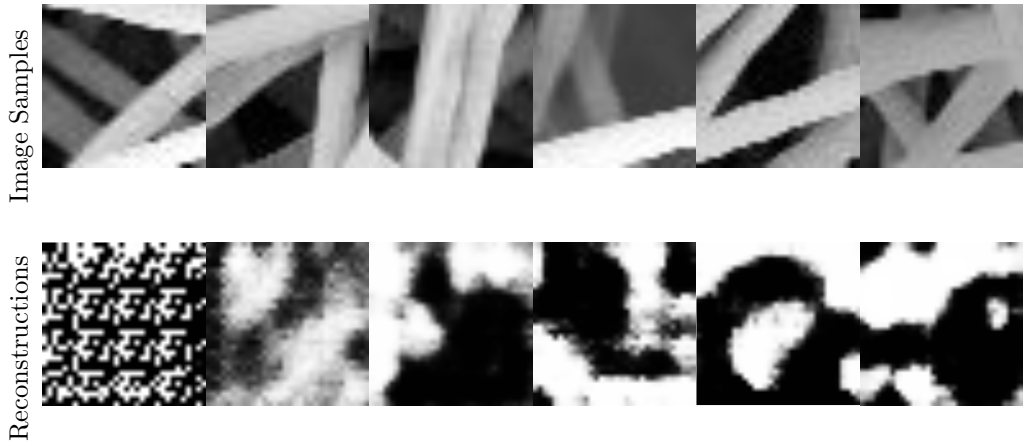


Figure 5.4 – Reconstruction examples from BiGAN model training with best configuration (Epochs 1,10,20,30,40 and 50)

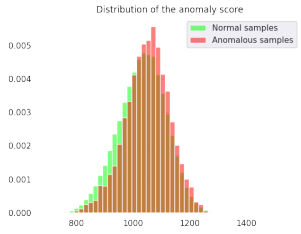
Using heuristics to improve the training proved useful in this model as the configuration with the best performance is the one with all three heuristics combined. Even with this boost, the model's precision and recall capacity is fairly limited as can be seen from the trade off graph in Figure 5.5. Even with the highest value of the threshold does not provide a precision above 50%.

Table 5.4
Ablation study for ALAD to test the effect of various training improvements for stabilization.

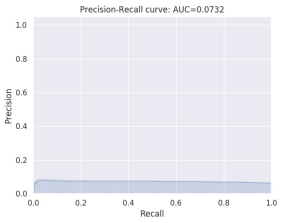
Model		**Metrics**			
		AUROC	Precision	Recall	F1 Score
ALAD	Normal	0.44842	0.05521	0.17443	0.08388
	IN	0.49447	0.06171	0.19385	0.09321
	SL	0.52920	**0.07854**	**0.24812**	**0.11931**
	LF	0.55545	0.07326	0.23146	0.11130
	IN + SL	0.48488	0.05918	0.18697	0.08990
	IN + LF	0.52712	0.06450	0.20379	0.09799
	SL + LF	0.49057	0.06441	0.20348	0.09784
	LF + SL + IN	**0.55904**	0.07588	0.23971	0.11527

Figure 5.6 shows the reconstruction capability of ALAD during training. Same as BiGAN, it suffers from the encoder, generator convergence

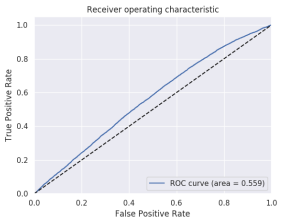
problem. There is no apparent visual resemblance between the input and the corresponding reconstructions.



(1) Separation Histogram



(2) Precision/Recall Trade off



(3) ROC Curve

Figure 5.5 – Visual performance analysis info for the best configuration of ablation study of ALAD

Figure 5.6 – Reconstruction examples from ALAD model training with best configuration (Epochs 1,10,20,30,40 and 50)

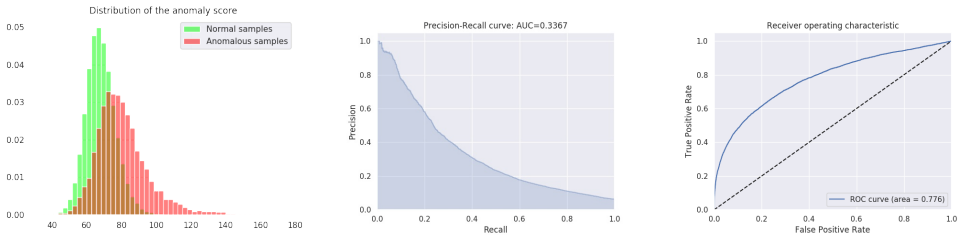
In table 5.5 we present the performance results for GANomaly. Since it is not a pure gan based model (along with Skip-GANomaly) their performance measure is not used as a comparison for the proposed model. However ablation study is performed to measure the impact of general performance because our proposed model procured its secondary encoder network with the initial intuition that the latent representation based anomaly detection score performs better than image space in GANomaly architecture.

Experiments for GANomaly shows that heuristics aimed to improve the GAN stabilization didn't improve the AUROC value obtained from the

Table 5.5
Ablation study for GANomaly to test the effect of various training improvements for stabilization.

Model		Metrics			
		AUROC	Precision	Recall	F1 Score
GANomaly	Normal	**0.77697**	**0.39701**	0.31356	**0.35038**
	IN	0.63222	0.09891	0.31249	0.15266
	SL	0.75224	0.35288	0.27870	0.31143
	LF	0.76225	0.37611	0.29704	0.33193
	IN + SL	0.62626	0.09693	0.30622	0.14725
	IN + LF	0.64009	0.10496	**0.33160**	0.15945
	SL + LF	0.76342	0.37495	0.29613	0.33091
	LF + SL + IN	0.63179	0.09548	0.30163	0.14504

baseline performance. One can conclude that because of the generator is created from an autoencoder architecture, there is not a convergence issue which observed on the pure GAN based models. As figure 5.7 shows, the separation histogram and precision/recall trade off presents a significantly better results compared to the prior ablation study results.



(1) Separation Histogram

(2) Precision/Recall Trade off

(3) ROC Curve

Figure 5.7 – Visual performance analysis info for the best configuration of ablation study of GANomaly

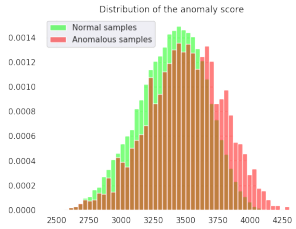
Table 5.6 shows the ablation study on the Skip-GANomaly model. All the combinations of heuristics marginally improved the overall AUROC performance. Despite noise inclusion obtained the best AUROC score, its combination with label flipping obtain a fairly close score with a slightly higher F1 score.

From the separation table and the precision/ recall trade off, we can observe that even though the Skip-GANomaly model is fairly similar to the GANomaly architecture, its performance is not better than AnoGAN

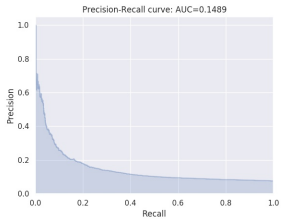
Table 5.6
Ablation study for Skip-GANomaly to test the effect of various training improvements for stabilization.

Model		Metrics			
		AUROC	Precision	Recall	F1 Score
Skip-GANomaly	Normal	0.53182	0.07379	0.23314	0.11211
	IN	**0.58120**	0.10646	0.25225	0.14973
	SL	0.55309	0.08686	0.27442	0.13196
	LF	0.53703	0.07617	0.24063	0.11571
	IN + SL	0.57432	0.10156	0.24063	0.14283
	IN + LF	0.58041	**0.12485**	0.19721	**0.15290**
	SL + LF	0.55675	0.08749	**0.27641**	0.13291
	LF + SL + IN	0.57402	0.10407	0.24659	0.14637

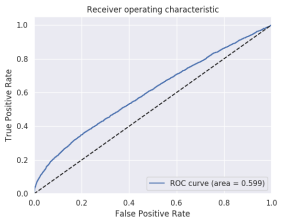
or BiGAN. We speculate that the main reason for this drop in the overall quality of the model is using image space to compute the anomaly score.



(1) Separation Histogram



(2) Precision/Recall Trade off



(3) ROC Curve

Figure 5.8 – Visual performance analysis info for the best configuration of ablation study of Skip-GANomaly

These experiments and ablation studies are done to gain insight towards the answers of two major questions.

- How does the GAN based anomaly detection frameworks present in the literature perform using our dataset ?
- If they don't perform well, is there a way to modify the aspect of the model that creates the problem to improve their performance ?

Our observations of the experiment results and their interpretation drew a path regards to the methodology we need to follow and the issues we need to focus. Following experiments will explain this approach.

5.4 Proposed Model Analysis

This section presents the experiments for our proposed model. Our methodology breaks down to three steps.

- Stabilizing the GAN training
- Solving convergence problem of encoder network for inverse mapping
- Experiments targeted to improve the performance

For the first step of the experimentation, energy based generative adversarial network is implemented and trained. The main focus of this experiment to observe whether generator network learns to create an artificial image which resembles the training data. Figure 5.9 shows the reconstruction samples of the generator from the EBGAN training.

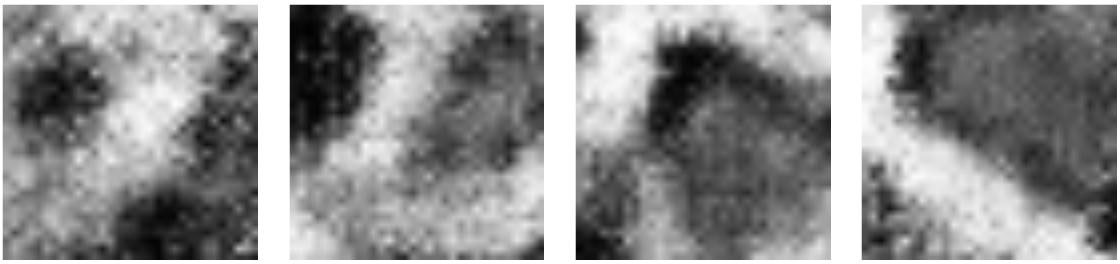
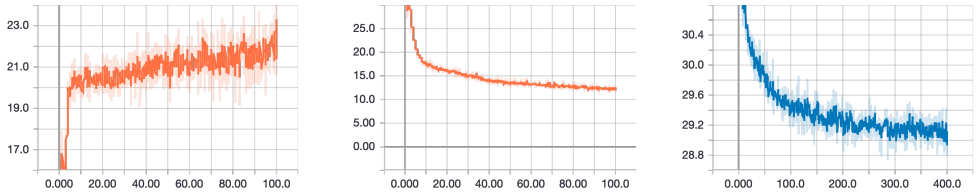


Figure 5.9 – Reconstruction samples from the EBGAN training

Second stage of the experiments is for the addition of encoder network and its sequential training. ENCEBGAN model (see section 4.3) is proposed to observe the reconstruction quality and anomaly detection performance. Figure 5.10 shows the training progression of the individual networks. Figure 5.11 presents the reconstruction samples both from training and inference. First two columns are obtained from the training of encoder network during training. The last column is the reconstruction of an anomalous sample from the testing phase.

Last stage of the experiments focus on improving the overall performance of the model. As Figure 5.13 shows, the reconstruction power of ENCEBGAN and SENCEBGAN models are similar in terms of the retention of the filament structure and its noisy nature. Latent space based anomaly detection methods are investigated to mitigate the effect of reconstruction quality to performance. Figure 5.12 presents the training process of the final proposed model. In the first stage, generator and discriminator models are trained with adversarial approach. In the second stage E_G (Generator Encoder) is trained using the fixed generator and discrimina-



(1) Generator Training Progression

(2) Discriminator Training Progression

(3) Encoder Training Progression

Figure 5.10 – Training progression of ENCEBGAN model

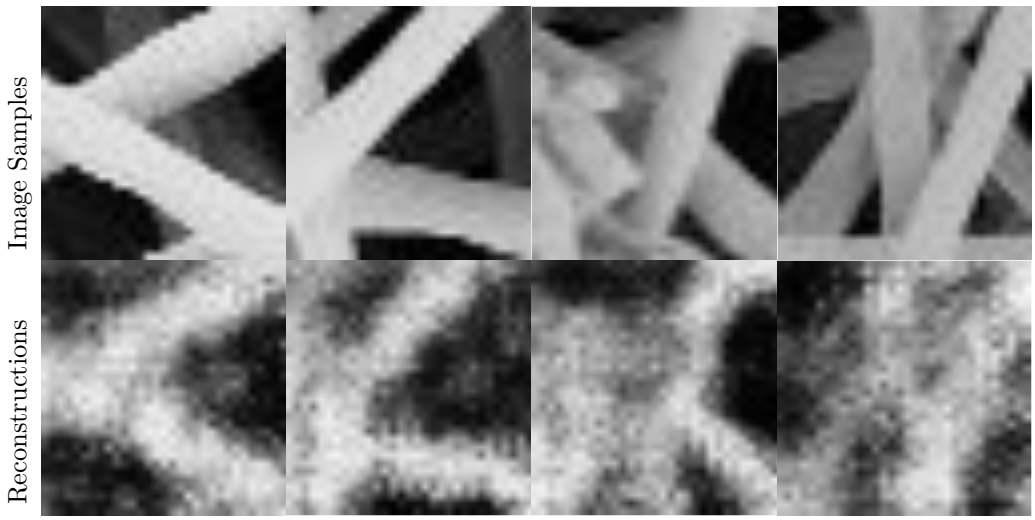
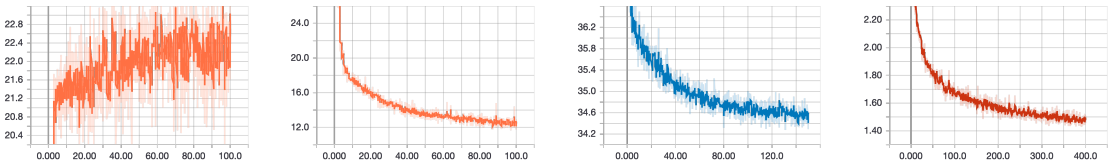


Figure 5.11 – Reconstruction examples of ENCEBGAN model obtained from both training and testing stage

tor networks. Finally in the third stage, E_R (Reconstruction Encoder) is trained by encoding the inputs and their reconstructions.



(1) Generator Training Progression

(2) Discriminator Training Progression

(3) Generator Encoder Training Progression

(4) Reconstruction Encoder Training Progression

Figure 5.12 – Training progression of SENCEBGAN model

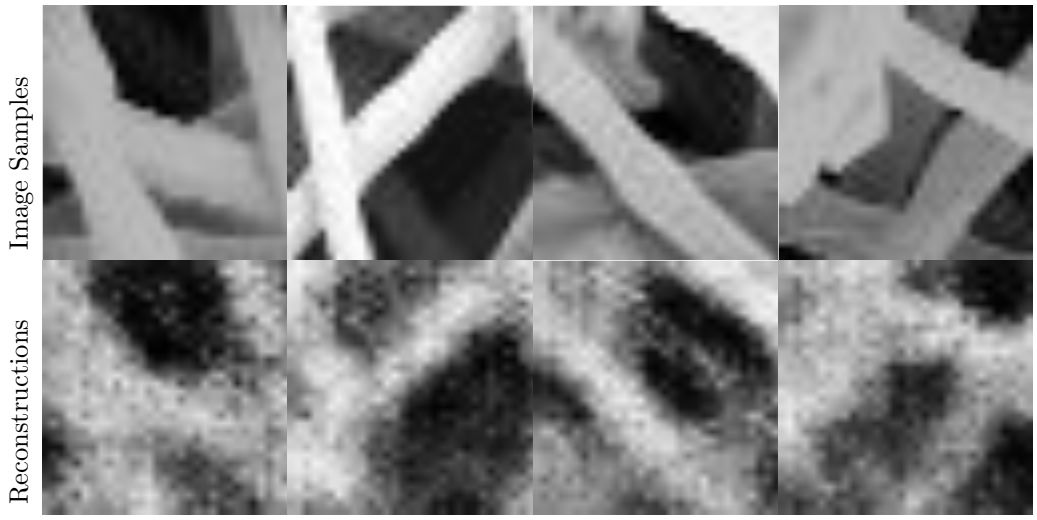


Figure 5.13 – Reconstruction examples of SENCEBGAN model obtained from both training and testing stage

Empirical analysis of all iterations of the proposed model is discussed in the next section.

5.5 Discussion of Experiment Results

Our motivation for these experiments was to create a model that successfully adapts generative adversarial network to learn the bidirectional mapping between the image and latent dimension space. Experiments performed on GAN based anomaly detection methods in the literature showed two main problems that affected performance.

- Generator network didn't succeed in learning the latent probability distribution p_{data} of the training dataset and couldn't generate visually similar images.
- Addition of encoder network to the adversarial training caused convergence issues hence reconstructions obtained by the encodings of the input images could not provide a reliable method for anomaly detection.

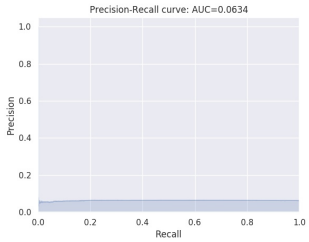
EBGAN model is proposed to provide a stable GAN training so that generator could learn to generate visually convincing samples. Comparison of Figure 5.9 with other models' generations (see Figures 5.4 and 5.6 for comparison) proves that visually our proposed GAN model learns to generate much more visually similar image samples.

Table 5.7
Performance comparison of our proposed model with other GAN based Anomaly detection models

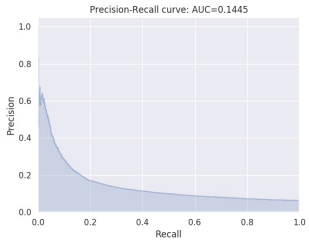
Models	Metrics			
	AUROC	Precision	Recall	F1 Score
AnoGAN	0.59941	0.14253	0.27783	0.18841
BiGAN	0.62699	0.10201	0.32227	0.15497
ALAD	0.44842	0.05521	0.17443	0.08388
EBGAN	0.50666	0.06341	0.06407	0.09733
ENCEBGAN	0.63190	0.14445	0.15379	0.18834
SENCEBGAN	0.71756	0.21142	0.21196	0.25958

ENCEBGAN model is proposed to improve the training of the encoder network. Method of back propagation AnoGAN (see 2.3.1) used was discarded mainly due to the computation time. To give a basic comparison, It took AnoGAN 7.2 hours to complete the testing phase while our model (and also other models that use encoder networks such as BiGAN and ALAD) completed testing phase in 5.1 minutes. Experiments regarding to the separate encoder training showed that not only we surpass the performance of other GAN based models, the reconstruction quality of the images in terms of retaining contextual information has improved greatly (see Figure 5.11). The problem encountered in this model iteration is that model didn't capture enough contextual information of some query images that contain many filaments spread across multiple depth layers. Our observations showed this is the main reason for majority of the false positive predictions. Last two columns of Figure 5.11 shows an example of this problem. The third column is a reconstruction of an anomalous sample and the last column is the reconstruction of a normal image. Due to the filament placement of the last image generator network could not reconstruct good enough version of the sample so contextual similariy is much less obvious compared to the reconstruction samples in the first two columns.

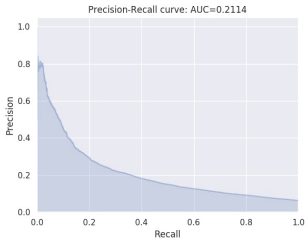
Second encoder network is added to our model to focus on latent dimension space of the images due to this problem. SENCEBGAN model outperforms other GAN based anomaly detection models with 14% increase in the AUROC value. During the experiments $\mathcal{L}_L(x)$ (see equation 4.8) is used as the main anomaly score computation method. Anomaly scores based on the latent representation obtained from the discriminator network didn't produced a notable result. Figure 5.14 shows the capacity comparison of our proposed model's iterations. First row shows the precision/ recall change with varying cut-off values. Second row shows the incremental change in



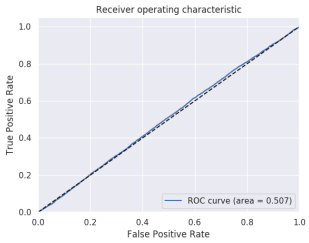
(1) EBGAN PRC Curve



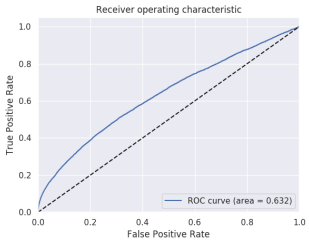
(2) ENCEBGAN PRC Curve



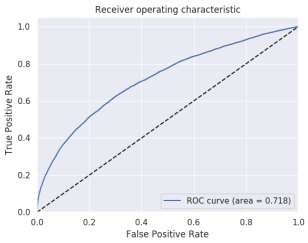
(3) SENCEBGAN PRC Curve



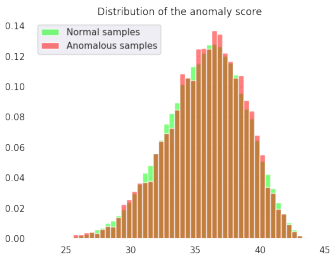
(4) EBGAN ROC Curve



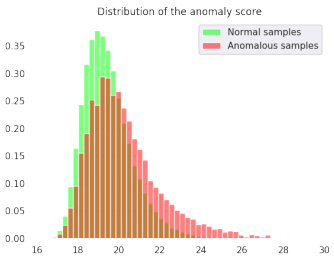
(5) ENCEBGAN ROC Curve



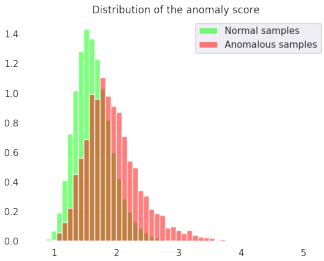
(6) SENCEBGAN ROC Curve



(7) EBGAN Separation Histogram



(8) ENCEBGAN Separation Histogram



(9) SENCEBGAN Separation Histogram

Figure 5.14 – Comparison of proposed model iterations using their PRC, ROC and Separation Histograms

ROC curve and bottom row presents the separation histogram of anomalous and normal samples in the dataset.

CHAPTER 6

Conclusion and Future Work

In this work, we tackled a problem of building an anomaly detection framework that utilizes a generative adversarial network and encoder networks. In particular, we investigated different GAN based anomaly detection and feature learning approaches and analyzed their corresponding performance with our SEM image dataset. Based on the analysis regarding the stabilization and convergence problems experienced by the networks, a modified architecture is proposed. In order to support our analyses, an ablation study is performed to mitigate the generator instabilities and the effect of training networks in an adversarial setting concurrently or sequentially is tested through an incremental model proposition.

Experiments presented in chapter 5 show that GAN based anomaly detection methods that are tested on the SEM Image dataset gives a contradictory performance scores with respect to the visual quality of the reconstructions. Even in model configurations which no convergence issue or mode collapse problem occurs, visual reconstruction imperfections cause model's performance to not pass a 0.6 threshold AUROC score. Models that inherit the adversarial training aspect of GANs but uses an autoencoder architecture in their generator networks produced better reconstructions and achieved a higher performance in terms of both AUROC score and the overall precision / recall capacity. These observations point out the importance of adversarial feature learning process that take effect in these models and how it affects the overall success of the anomaly detection quality.

Proposed model which addresses the issues detected in the experiments produced a better performance than the other GAN based models. Reconstructions obtained by the model showed that separating the networks

to stabilize the training procedure improved the visual resemblance while there is still a room to improve for the quality of the reconstructions. In particular noisy reconstructions of some of the patches has a high chance of being identified as an anomalous sample. Observation of autoencoder variant models also showed that computing the anomaly representation using latent dimensional space instead of the image space can decrease the impact of this problem. Model proposed based on this observations produced an AUROC score above 0.7 in the experiments. We speculate that main factor to further improve this approach also depends on the quality of the reconstructions.

As the future work which can be built on top the results of this thesis, it is possible to investigate further open issues. Energy based generative adversarial networks are relatively new topic and the quality of the reconstructions has a potential for improvement. Functionality of the autoencoder based discriminator network can also be extended to learn the joint distribution of the image and latent representation like in BiGAN (see section 2.3.2) to investigate how energy based adversarial loss impact on the concurrent learning of bidirectional mapping.

Moreover, acquiring a relevant data for a problem is still an important issue in all types of machine learning task. Apart from its generation capabilities and its contribution to the adversarial feature learning process, GANs and its variational autoencoder variants can be used for the data augmentation task. Particularly [38] proposes an oversampling method for infrequent normal samples which focuses on to improve the false positive rate for unsupervised anomaly detection tasks.

APPENDIX A

Model Implementation Details

This section contains all the model implementation details. The repository for the thesis can be found at this address. All the experiments are conducted with the corresponding configuration files in the **configs** folder. Tensorflow 1.13[1] is used as the underlying framework with Python 3.7[2]

A.1 AnoGAN Implementation

Table A.1
Architecture and Hyperparameters of AnoGAN Model

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Generator					
Dense			$4 \times 4 \times 512$	✓	Leaky Relu
Transposed Convolution	4×4	2×2	512	✓	Leaky Relu
Transposed Convolution	4×4	2×2	256	✓	Leaky Relu
Transposed Convolution	4×4	2×2	128	✓	Leaky Relu
Transposed Convolution	5×5	1×1	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5e-5,$ beta $1 = 0.5,$ beta $2 = 0.999,$)				
Discriminator					
Convolution	4×4	2×2	64	✓	Leaky Relu
Convolution	4×4	2×2	128	✓	Leaky Relu
Convolution	4×4	2×2	256	✓	Leaky Relu

[1] https://www.tensorflow.org
[2] https://www.python.org/downloads/release/python-370/

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Dense			1		Sigmoid
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,$ beta $1 = 0.5,$ beta $2 = 0.999,$)				
Epochs	50				
Batch Size	64				

A.2 BiGAN Implementation

Table A.2
Architecture and Hyperparameters of BiGAN Model

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Generator					
Dense			$4 \times 4 \times 512$	✓	Leaky Relu
Transposed Convolution	4×4	2×2	512	✓	Leaky Relu
Transposed Convolution	4×4	2×2	256	✓	Leaky Relu
Transposed Convolution	4×4	2×2	128	✓	Leaky Relu
Transposed Convolution	×5	1×1	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-5,$ beta $1 = 0.5,$ beta $2 = 0.999,$)				
Discriminator					
Convolution	4×4	2×2	64	✓	Leaky Relu
Convolution	4×4	2×2	128	✓	Leaky Relu
Convolution	4×4	2×2	256	✓	Leaky Relu
Dense			1		Sigmoid
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,$ beta $1 = 0.5,$ beta $2 = 0.999,$)				
Encoder					
Convolution	4×4	2×2	64		Leaky Relu
Convolution	4×4	2×2	128	✓	Leaky Relu
Convolution	4×4	2×2	256	✓	Leaky Relu
Dense			256		
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,$ beta $1 = 0.5,$ beta $2 = 0.999,$)				
Epochs	50				
Batch Size	64				

A.3 ALAD Implementation

Table A.3
Architecture and Hyperparameters of ALAD Model

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Generator					
Dense			$4 \times 4 \times 512$	✓	Leaky Relu
Transposed Convolution	4×4	2×2	512	✓	Leaky Relu
Transposed Convolution	4×4	2×2	256	✓	Leaky Relu
Transposed Convolution	4×4	2×2	128	✓	Leaky Relu
Transposed Convolution	4×4	1×1	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-5,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Discriminator XZ					
Convolution(x)	4×4	2×2	128	✓	Leaky Relu
Convolution(x)	4×4	2×2	256	✓	Leaky Relu
Convolution(x)	4×4	2×2	512	✓	Leaky Relu
Reshape	Batch $\times 512 \times 4 \times 4$				
Convolution(z)	4×4	2×2	512	Dropout	Leaky Relu
Convolution(z)	4×4	2×2	512	Dropout	Leaky Relu
Concatenate					
Convolution	1×1	1×1	1024	Dropout	Leaky Relu
Convolution	1×1	1×1	1024	Dropout	Leaky Relu
Dense			1		Sigmoid
Leaky ReLU Slope	0.2				
Dropout Rate	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Encoder					
Convolution	4×4	2×2	32	✓	Leaky Relu
Convolution	4×4	2×2	64	✓	Leaky Relu
Convolution	4×4	2×2	128	✓	Leaky Relu
Convolution	4×4	2×2	256	✓	Leaky Relu
Convolution	4×4	2×2	256		
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Discriminator XX					
Concatenate	f				
Convolution	4×4	2×2	64	Dropout	Leaky Relu
Convolution	4×4	2×2	128	Dropout	Leaky Relu
Dense			1		Sigmoid
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Discriminator ZZ					
Dense	4×4	2×2	64	Dropout	Leaky Relu

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Dense	4×4	2×2	32	Dropout	Leaky Relu
Dense	4×4	2×2	1	Dropout	Leaky Relu
Leaky ReLU Slope	0.2				
Dropout Rate	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Epochs	50				
Batch Size	64				

A.4 GANomaly Implementation

Table A.4
Architecture and Hyperparameters of GANomaly Model

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Generator					
Encoder					
Convolution	5×5	2×2	64		Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Convolution	5×5	2×2	256	✓	Leaky Relu
Dense			256		
Decoder					
Transposed Convolution	4×4	2×2	256	✓	Leaky Relu
Transposed Convolution	4×4	2×2	128	✓	Leaky Relu
Transposed Convolution	4×4	2×2	64	✓	Leaky Relu
Transposed Convolution	4×4	2×2	32	✓	Leaky Relu
Transposed Convolution	4×4	2×2	16	✓	Leaky Relu
Transposed Convolution	4×4	1×1	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-5,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Encoder 2					
Convolution	5×5	2×2	64		Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Convolution	5×5	2×2	256	✓	Leaky Relu
Dense			256		
Discriminator					
Convolution	4×4	2×2	64	✓	Leaky Relu
Convolution	4×4	2×2	128	✓	Leaky Relu
Convolution	4×4	2×2	256	✓	Leaky Relu
Dense			1		Sigmoid
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Epochs	50				
Batch Size	64				

A.5 Skip-GANomaly Implementation

Table A.5
Architecture and Hyperparameters of Skip-GANomaly Model

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Generator					
Convolution	5×5	2×2	64	✓	Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Convolution	5×5	2×2	256	✓	Leaky Relu
Convolution	5×5	2×2	512	✓	Leaky Relu
Convolution	5×5	2×2	512	✓	Leaky Relu
Transposed Convolution	5×5	2×2	512	✓	Leaky Relu
Transposed Convolution	5×5	2×2	256	✓	Leaky Relu
Transposed Convolution	5×5	2×2	128	✓	Leaky Relu
Transposed Convolution	5×5	2×2	64	✓	Leaky Relu
Transposed Convolution	5×5	2×2	1		
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-5,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Discriminator					
Convolution	5×5	2×2	64	✓	Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Convolution	5×5	2×2	256	✓	Leaky Relu
Dense			1		Sigmoid
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Epochs	50				
Batch Size	64				

A.6 ENCEBGAN Implementation

Table A.6
Architecture and Hyperparameters of ENCEBGAN Model

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Generator					

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Dense			$4 \times 4 \times 256$	✓	Leaky Relu
Transposed Convolution	5×5	2×2	128	✓	Leaky Relu
Transposed Convolution	5×5	2×2	64	✓	Leaky Relu
Transposed Convolution	5×5	2×2	32	✓	Leaky Relu
Transposed Convolution	5×5	2×2	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-5,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Discriminator					
Convolution	5×5	2×2	32	✓	Leaky Relu
Convolution	5×5	2×2	64	✓	Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Dense			256		
Transposed Convolution	5×5	2×2	256	✓	Leaky Relu
Transposed Convolution	5×5	2×2	128	✓	Leaky Relu
Transposed Convolution	5×5	2×2	64	✓	Leaky Relu
Transposed Convolution	5×5	2×2	32	✓	Leaky Relu
Transposed Convolution	5×5	2×2	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-6,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Encoder					
Convolution	5×5	2×2	64		Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Convolution	5×5	2×2	256	✓	Leaky Relu
Dense			256		
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-5,\ \mathrm{beta}\ 1 = 0.5,\ \mathrm{beta}\ 2 = 0.999,$)				
Epochs	50				
Batch Size	64				

A.7 SENCEBGAN Implementation

Table A.7
Architecture and Hyperparameters of SENCEBGAN Model

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Generator					
Dense			$4 \times 4 \times 256$	✓	Leaky Relu
Transposed Convolution	5×5	2×2	128	✓	Leaky Relu
Transposed Convolution	5×5	2×2	64	✓	Leaky Relu
Transposed Convolution	5×5	2×2	32	✓	Leaky Relu

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Transposed Convolution	5×5	2×2	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-5$, beta $1 = 0.5$, beta $2 = 0.999$,)				
Discriminator					
Convolution	5×5	2×2	32	✓	Leaky Relu
Convolution	5×5	2×2	64	✓	Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Dense			256		
Transposed Convolution	5×5	2×2	256	✓	Leaky Relu
Transposed Convolution	5×5	2×2	128	✓	Leaky Relu
Transposed Convolution	5×5	2×2	64	✓	Leaky Relu
Transposed Convolution	5×5	2×2	32	✓	Leaky Relu
Transposed Convolution	5×5	2×2	1		TanH
Latent Dimension	256				
Leaky Relu Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-6$, beta $1 = 0.5$, beta $2 = 0.999$,)				
Encoder					
Convolution	5×5	2×2	64		Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Convolution	5×5	2×2	256	✓	Leaky Relu
Dense			256		
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-5$, beta $1 = 0.5$, beta $2 = 0.999$,)				
Encoder 2					
Convolution	5×5	2×2	64		Leaky Relu
Convolution	5×5	2×2	128	✓	Leaky Relu
Convolution	5×5	2×2	256	✓	Leaky Relu
Dense			256		
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 5\mathrm{e}-5$, beta $1 = 0.5$, beta $2 = 0.999$,)				
Discriminator XX					
Concatenate	f				
Convolution	4×4	2×2	64	Dropout	Leaky Relu
Convolution	4×4	2×2	128	Dropout	Leaky Relu
Dense			1		Sigmoid
Leaky ReLU Slope	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6$, beta $1 = 0.5$, beta $2 = 0.999$,)				
Discriminator ZZ					
Dense	4×4	2×2	64	Dropout	Leaky Relu
Dense	4×4	2×2	32	Dropout	Leaky Relu

Operation	Kernel	Stride	Feature Maps/ Units	BN ?	Activation
Dense	4×4	2×2	1	Dropout	Leaky Relu
Leaky ReLU Slope	0.2				
Dropout Rate	0.2				
Batch Norm Momentum	0.8				
Optimizer	Adam ($\mathrm{lr} = 1\mathrm{e}-6,$ beta $1 = 0.5,$ beta $2 = 0.999,$)				
Epochs	50				
Batch Size	64				

Bibliography

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. "A survey of network anomaly detection techniques". In: *Journal of Network and Computer Applications* 60 (2016), pp. 19 –31. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2015.11.016. URL: http://www.sciencedirect.com/science/article/pii/S1084804515002891.

[2] Samet Akçay, Amir Atapour Abarghouei, and Toby P. Breckon. "GANomaly: Semi-supervised Anomaly Detection via Adversarial Training". In: *ACCV*. 2018.

[3] Samet Akçay, Amir Atapour Abarghouei, and Toby P. Breckon. "Skip-GANomaly: Skip Connected and Adversarially Trained Encoder-Decoder Anomaly Detection". In: *CoRR* abs/1901.08954 (2019).

[4] Jinwon An and Sungzoon Cho. "Variational autoencoder based anomaly detection using reconstruction probability". In: *Special Lecture on IE* 2 (2015), pp. 1–18.

[5] Plamen Angelov, Ramin Ramezani, and Xiaowei Zhou. "Autonomous Novelty Detection and Object Tracking in Video Streams using Evolving Clustering and Takagi-Sugeno type Neuro-Fuzzy System". In: June 2008, pp. 1456–1463. DOI: 10.1109/IJCNN.2008.4633989.

[6] Martin Arjovsky and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *stat* 1050 (Jan. 2017).

[7] Martín Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *CoRR* abs/1701.07875 (2017).

[8] Pierre Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012, pp. 37–49.

[9] M. Behniafar, A.R. Nowroozi, and H.R. Shahriari. "A Survey of Anomaly Detection Approaches in Internet of Things". In: *The ISC International Journal of Information Security* 10.2 (2018), pp. 79–92. ISSN: 2008-2045. DOI: 10.22042/isecure.2018.116976.408. eprint: http://www.isecure-journal.com/article___835f5ca77e495cec328b957653e1f4d566995.pdf. URL: http://www.isecure-journal.com/article_66995.html.

[10] Yoshua Bengio. "Learning Deep Architectures for AI". In: *Found. Trends Mach. Learn.* 2.1 (Jan. 2009), pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: http://dx.doi.org/10.1561/2200000006.

[11] Patrick Billingsley. *Probability and Measure*. Second. John Wiley and Sons, 1986.

[12] Giacomo Boracchi, Diego Carrera, and Brendt Wohlberg. "Novelty detection in images by sparse representations". In: *2014 IEEE Symposium on Intelligent Embedded Systems (IES)*. IEEE. 2014, pp. 47–54.

[13] Diego Carrera et al. "Defect detection in SEM images of nanofibrous materials". In: *IEEE Transactions on Industrial Informatics* 13.2 (2016), pp. 551–561.

[14] Diego Carrera et al. "Scale-invariant anomaly detection with multiscale group-sparse models". In: *Proceedings of IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, USA, Sept. 2016, pp. 3892–3896. DOI: 10.1109/ICIP.2016.7533089.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009), 15:1–15:58. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: http://doi.acm.org/10.1145/1541880.1541882.

[16] Lei Clifton et al. "Identification of Patient Deterioration in Vital-Sign Data using One-Class Support Vector Machines." In: Jan. 2011, pp. 125–131.

[17] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006. ISBN: 0471241954.

[18] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial Feature Learning". In: *CoRR* abs/1605.09782 (2017).

[19] Vincent Dumoulin et al. "Adversarially Learned Inference". In: *CoRR* abs/1606.00704 (2017).

[20] Gilberto Fernandes et al. “Network anomaly detection using IP flows with Principal Component Analysis and Ant Colony Optimization”. In: *Journal of Network and Computer Applications* 64 (2016), pp. 1 –11. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2015.11.024. URL: http://www.sciencedirect.com/science/article/pii/S1084804516000618.

[21] Ugo Fiore et al. “Using Generative Adversarial Networks for Improving Classification Effectiveness in Credit Card Fraud Detection”. In: *Information Sciences* (Dec. 2017). DOI: 10.1016/j.ins.2017.12.030.

[22] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *AISTATS*. 2011.

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

[24] Ian J. Goodfellow et al. “Generative Adversarial Nets”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’14. Montreal, Canada: MIT Press, 2014, pp. 2672–2680. URL: http://dl.acm.org/citation.cfm?id=2969033.2969125.

[25] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. In: *NIPS*. 2017.

[26] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[27] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5967–5976.

[28] J. Jabez and B. Muthukumar. “Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach”. In: *Procedia Computer Science* 48 (2015). International Conference on Computer, Communication and Convergence (ICCC 2015), pp. 338 –346. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.04.191. URL: http://www.sciencedirect.com/science/article/pii/S1877050915007000.

[29] Tony Jebara and Marina Meila. “Machine learning: Discriminative and generative”. In: *The Mathematical Intelligencer* 28 (Mar. 2006), pp. 67–69. DOI: 10.1007/BF02987011.

[30] Dongil Kim et al. “Machine Learning-based Novelty Detection for Faulty Wafer Detection in Semiconductor Manufacturing”. In: *Expert Syst. Appl.* 39.4 (Mar. 2012), pp. 4075–4083. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2011.09.088. URL: http://dx.doi.org/10.1016/j.eswa.2011.09.088.

[31] Taeksoo Kim et al. “Learning to Discover Cross-Domain Relations with Generative Adversarial Networks”. In: *ICML*. 2017.

[32] Walter Kramer. “Probability & Measure : Patrick Billingsley (1995): (3rd ed.). New York : Wiley, ISBN 0-471-0071-02, pp 593, [pound sign] 49.95”. In: *Computational Statistics & Data Analysis* 20.6 (1995), pp. 702–703. URL: https://ideas.repec.org/a/eee/csdana/v20y1995i6p703-702.html.

[33] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. “CIFAR-10 (Canadian Institute for Advanced Research)”. In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[34] Anders Boesen Lindbo Larsen et al. “Autoencoding beyond pixels using a learned similarity metric”. In: *ICML*. 2016.

[35] Yann LeCun et al. “A tutorial on energy-based learning”. In: *PREDICTING STRUCTURED DATA*. MIT Press, 2006.

[36] Valentin Leveau and Alexis Joly. “Adversarial autoencoders for novelty detection”. In: (2017).

[37] Chunyuan Li et al. “Towards Understanding Adversarial Learning for Joint Distribution Matching”. In: *NIPS*. 2017.

[38] Swee Kiat Lim et al. “DOPING: Generative Data Augmentation for Unsupervised Anomaly Detection with GAN”. In: *CoRR* abs/1808.07632 (2018). arXiv: 1808.07632. URL: http://arxiv.org/abs/1808.07632.

[39] Markos Markou and Sameer Singh. “Novelty Detection: A Review&Mdash;Part 2: Neural Network Based Approaches”. In: *Signal Process.* 83.12 (Dec. 2003), pp. 2499–2521. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2003.07.019. URL: http://dx.doi.org/10.1016/j.sigpro.2003.07.019.

[40] Luis Martí et al. “Anomaly Detection Based on Sensor Data in Petroleum Industry Applications”. In: *Sensors* 15.2 (2015), pp. 2774–2797. ISSN: 1424-8220. DOI: 10.3390/s150202774. URL: http://www.mdpi.com/1424-8220/15/2/2774.

[41] Takeru Miyato et al. “Spectral Normalization for Generative Adversarial Networks”. In: Feb. 2018.

[42] Paolo Napoletano, Flavio Piccoli, and Raimondo Schettini. "Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity". In: *Sensors* 18.2 (2018), p. 209. ISSN: 1424-8220. DOI: 10.3390/s18010209. URL: http://dx.doi.org/10.3390/s18010209.

[43] Yuval Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: 2011.

[44] Stanislav Pidhorskyi et al. "Generative Probabilistic Novelty Detection with Adversarial Autoencoders". In: *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 6823–6834. URL: http://dl.acm.org/citation.cfm?id=3327757.3327787.

[45] Marco A. F. Pimentel et al. "Review: A Review of Novelty Detection". In: *Signal Process.* 99 (June 2014), pp. 215–249. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2013.12.026. URL: http://dx.doi.org/10.1016/j.sigpro.2013.12.026.

[46] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *CoRR* abs/1511.06434 (2016).

[47] Douglas A. Reynolds. "Gaussian Mixture Models". In: *Encyclopedia of Biometrics*. 2009.

[48] Bengt Ringnér. "Law of the unconscious statistician". In: (2009).

[49] George G Roussas. *An introduction to measure-theoretic probability*. Burlington, MA: Elsevier, 2004. URL: http://cds.cern.ch/record/1614120.

[50] Mohammad Sabokrou et al. *Adversarially Learned One-Class Classifier for Novelty Detection*. cite arxiv:1802.09088Comment: CVPR 2018 Accepted Paper. 2018. URL: http://arxiv.org/abs/1802.09088.

[51] Ruslan Salakhutdinov and Geoffrey Hinton. "Deep Boltzmann Machines". In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, pp. 448–455. URL: http://proceedings.mlr.press/v5/salakhutdinov09a.html.

[52] Ruslan R. Salakhutdinov and Geoffrey E. Hinton. "Deep Boltzmann Machines". In: *AISTATS*. 2009.

[53] Tim Salimans et al. "Improved Techniques for Training GANs". In: (June 2016).

[54] Thomas Schlegl et al. “f-AnoGAN: Fast Unsupervised Anomaly Detection with Generative Adversarial Networks”. In: *Medical Image Analysis* 54.Data Mining and Knowledge Discovery 29 3 2015 (2019), pp. 30–44. DOI: 10.1016/j.media.2019.01.010. URL: https://app.dimensions.ai/details/publication/pub.1111824956.

[55] Thomas Schlegl et al. “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery”. In: *IPMI*. 2017.

[56] “SEM Image Dataset)”. In: (). URL: http://web.mi.imati.cnr.it/ettore/NanoTwice.

[57] H.H. Sohrab. *Basic Real Analysis*. Birkhäuser Boston, 2011. ISBN: 9780817682323. URL: https://books.google.it/books?id=zjfaBwAAQBAJ.

[58] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[59] Raihan Ul Islam, Mohammad Shahadat Hossain, and Karl Andersson. “A novel anomaly detection algorithm for sensor data under uncertainty”. In: *Soft Computing* 22.5 (2018), pp. 1623–1639. ISSN: 1433-7479. DOI: 10.1007/s00500-016-2425-2. URL: https://doi.org/10.1007/s00500-016-2425-2.

[60] Bernard Widrow and Eugene Walach. “Appendix G: Thirty Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation”. In: 2008.

[61] Zili Yi et al. “DualGAN: Unsupervised Dual Learning for Image-to-Image Translation”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2868–2876.

[62] Houssam Zenati et al. “Adversarially Learned Anomaly Detection”. In: *CoRR* abs/1812.02288 (2018). arXiv: 1812.02288. URL: http://arxiv.org/abs/1812.02288.

[63] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. “Energy-based Generative Adversarial Network”. In: *CoRR* abs/1609.03126 (2016).

[64] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2242–2251.

[65] Maria Zontak and Israel Cohen. “Defect detection in patterned wafers using anisotropic kernels”. In: *Machine Vision and Applications* 21.2 (2010), pp. 129–141.