CITY, UNIVERSITY OF LONDON

MSc in Data Science

Project Report

December 2022

CAR DAMAGE DETECTION VIA UNSUPERVISED GENERATIVE ADVERSARIAL NETWORKS

Author: Elnara Mammadova

Project Report Supervised by: Kevin Ryan

External Supervisor: Toby Staines

Submitted: 11th of January, 2023

DECLARATION

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work, I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed: ***Elnara Mammadova***

ABSTRACT

The purpose of this project was to solve for anomaly detection problem via an adversarial learning process using a real-world image dataset. The study was focused on the prospect of car damage detection based on the reconstruction of uncorrupted clean samples from any arbitrary image through an unsupervised Generative Adversarial Network (GAN) training process. This work studies car damage detection by researching over two baseline model architectures: GANomaly (Akcay et al., 2018) and SkipGANomaly (Akçay et al., 2019). The aim of this research was to test both models on a much more complex real-world image dataset and improve upon the results to add to the body of knowledge. The research was able to produce a variant of traditional GANomaly methods via the use of advanced training techniques referenced in the previous research literature. Most importantly, a new variant of the SkipGANomaly method was proposed by iteratively projecting arbitrary input towards the clean distribution in the target domain through the introduction of synthetic anomalies in the form of masked regions. By reformulating the original reconstruction task of the SkipGANomaly method as an image completion problem, we were able to produce unprecedented results in successfully completing the masked image regions with the anomaly-free versions. We were able to demonstrate the relevance of our approach with a consistent assessment of the reconstruction-based image completion method, by comparing its performance over a complex client dataset. This study provides substantial proof of concept demonstrating the potential GAN frameworks have to offer towards unsupervised anomaly detection in complex and multifaceted image datasets.

Keywords: Generative Adversarial Learning, Anomaly Detection, GANomaly, SkipGANomaly, Image Reconstruction, Deep Neural Networks

ACKNOWLEDGEMENTS

I would like to thank many people who contributed and supported me in reaching this goal. First and foremost, I would like to thank my supervisor, Kevin Ryan, who has contributed much to this research, and for being patient, helpful, and encouraging. I especially would like to thank my external supervisor Toby Staines for providing the data and support for this research through an internship with Tractable AI. I also thank all the researchers at Tractable AI, most importantly Martha Robinson, Shuyu Lin, and Chris Lucas for being a constant source of productive and thoughtful feedback.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at the City, University of London a great experience. Special thanks to my small group of friends at the university, namely, Alessandro Alviani, Dimitris Megkos, Jerome Titus and Anglina Bhambra for their unwavering support through tough times and unforgettable memories that will carry with me for a long time.

I would like to express my deepest appreciation to my husband, Mustafa Ayyuru, who has always been supportive and encouraging of my accomplishments. However, I unequivocally dedicate my success in completing this degree to my 2-year-old daughter, Dilayla Mammadova Ayyuru. This accomplishment would not have been possible without her unwavering love and smile.

Lastly, I would like to thank my dog, Misha, who passed away last year, for teaching me that when the going gets tough, it's time to go for a walk outside to clear your mind.

TABLE OF CONTENTS

1 INTRODUCTION AND OBJECTIVES

1.1 Introduction

Recent research and progress in Deep Learning have led to significant advancements in Machine Learning, making it possible for some algorithms to outperform even humans in certain tasks. With the vast growth of data, much of it readily accessible, truly understanding and discovering patterns in this treasure trove of data is one of the main challenges for these algorithms. One of the most promising approaches towards achieving this goal is generative models. *Generative modelling* is an unsupervised or semi-supervised machine learning technique that learns the natural patterns in input data to create novel data samples that are very similar or analogous to the original dataset. The famous quote by Richard Feynman is the best example to illustrate the intuition behind generative models: "What I cannot create, I do not understand." In the race to replicate human intelligence, Generative Adversarial Networks (Goodfellow et al., 2014), or GANs for short, have emerged as the front-runner among other generative models. GANs have prevailed in generating novel, high-resolution data samples such as images of non-existent people, animals, objects, etc., as well as audio and text. Alternatively, due to their success in modelling complex and high dimensional distributions, GANs have also become an important research field in anomaly detection. In this thesis, we will attempt to detect anomalies (damages) in car images by employing GANs and the adversarial learning process.

Anomaly detection is the task of identifying abnormal patterns in the data that deviate or differ significantly from the majority of instances and do not conform to a well-defined notion of normal behaviour (Chandola et al., 2009). The ability of GANs to solve the problem of anomaly detection without having any prior knowledge of the anomalies is what makes them superior to other supervised and semi-supervised techniques. The purpose of this research is to learn the normality of the data via unsupervised GAN learning and attempt to reconstruct the anomalous images as normal instances from the learned data distribution during inference time, and subsequently visualize the damages via the mask difference between the damaged set and the generated undamaged images.

This is a Master's Thesis project for the MSc program in Data Science offered by the Department of Computer Science at City, University of London, supervised by department lecturer Kevin Ryan[1]. As part of a client-based internship project for Tractable AI[2], a technology company that develops Artificial

[1] Kevin Ryan, Lecturer of Data Science at City, University of London (kevin.ryan@city.ac.uk)

[2] Tractable AI is a technology company aiming to provide industry solutions for auto and property claims through the capabilities the Artificial Intelligence has to offer (https://tractable.ai/en)

Intelligence (AI) solutions for visual damage assessment, this project was also supervised externally by Toby Staines[3].

1.2 Problem Background

Since the recent advances in Artificial Intelligence (AI), many industries have automated most of their production and services, making them more efficient by improving the speed of delivery with minimal human intervention. Auto insurance companies are one of those industries with a large scope for automation. Generally, auto-claims can be lengthy and expensive, particularly if there has been a claims leakage. Claims leakage refers to the difference between the amount of claim settlement agreed upon by the insurer and the amount they could have paid had there been a more effective claim process. Depending on the complexity of the claims, each claim may require scores of expert knowledge and involvement through to the final settlement. Consequently, it can necessitate substantial administrative overhead for visual inspection and validation of vehicle damages, as well as mistakes made along the way that may hold up the process and increase the cost and duration of the claims. In addition, due to the subjectivity of human evaluators, discrepancies in individual experts' judgement might result in inconsistent and often erroneous claim processes. To both expedite the delivery of auto-claim services and supply consistent and reliable estimates, many auto insurance companies have adopted computer vision technologies through deep neural network methods. An automated system of auto insurance claims can help eliminate human error caused by manual processing and allow real-time monitoring of claims, preventing additional costs to accrue.

As part of the Postgraduate Internship Scheme within the City, University of London, this project is based on the work undertaken as part of an internship project with Tractable AI. Tractable is one of the leading pioneers in the industry of technology companies developing Artificial Intelligence (AI) for vehicle damage assessment. In the past, the company has successfully provided automated damage detection solutions using state-of-art supervised and semi-supervised learning methods, which generally required laborious and expensive labelling and segmentation tasks. In an effort to lessen their reliance on labelled datasets, researchers at Tractable are continuously looking into improving their car damage detection workflows using unsupervised deep learning methods.

This project aims to answer the following question:

Can Generative Adversarial Networks (GANs) help detect car damage via unsupervised learning?

[3] Toby Staines is a Senior Applied Researcher at Tractable AI (toby.staines@tractable.ai)

1.3 Objectives and beneficiaries

In a quest for reducing the need for labelled datasets, there is a plethora of unsupervised methods to choose from, designed specifically for deep neural networks. These methods have been very effective in improving generalisation performance using only unlabelled datasets. GANs as an unsupervised model have achieved great success in many hard computer vision tasks, more so in recent years. One of the primary benefits of GAN is its capacity to learn the underlying structure of unlabelled, complex, and messy datasets and transform it to reproduce the learned distribution, thereby generating novel synthetic images. Tractable AI, having access to a tremendous amount of data through their clients, which are mostly unlabelled and unstructured, faces many challenges when it comes to damage detection without undertaking laborious and extensive labelling efforts. This project deeply reviews the theoretical basis of GANs and surveys some recently developed unsupervised GAN frameworks in anomaly detection.

In order to answer the question posed by this research project mentioned in the previous section, this project has three key objectives that it needs to accomplish:

1) Aim to detect anomalies in real-world, unlabelled and messy image datasets without the use of any new information using GAN frameworks.
2) Using the recent GAN frameworks generate anomaly-free versions of damaged car images via the learned distribution of uncorrupted clean samples.
3) Provide a proof of concept for future work demonstrating the potential GANs has to offer in unsupervised anomaly detection that can be further researched by Tractable AI with a more curated and larger dataset and deeper networks.

The beneficiaries of this work will be Tractable AI, as this research provides the potential and the drawbacks of using unsupervised GAN frameworks in anomaly detection, which will allow them to decide whether they will want to carry out more extensive research towards this area. Additionally, the models curated through this project will allow them to test some of their hypotheses immediately, and subsequently build on it through comprehensive improvements. Outside Tractable AI the research and the model templates could be used by anyone wanting to detect anomalies in other problem domains and get a model up and running for quick solutions.

1.4 Work Plan

The structure of this work is based on the work plan shown in Figure 1.1. The project kicks off with an extensive context and literature review of published works related to anomaly detection through unsupervised GAN frameworks. Initially, the two proposed frameworks studied in this project were not part of the original research proposal, as they became the main focus of this work through a more

extensive literature study during the first stage of this project. The next stage of the project involves data manipulation, Exploratory Data Analysis (EDA) and data pre-pre-processing for a robust data pipeline in order to keep the computational cost low. After the model selection of the two most recent and state-of-art GAN frameworks in anomaly detection, we proceeded with reproducing the model architectures which are publicly available on GitHub[4] in PyTorch[5] library. To gain a deeper familiarity with the models and more leeway for customization, we programmed the model architecture from scratch using TensorFlow[6] and Keras[7] libraries. The main part of the project consisted of the implementation and evaluation of two frameworks, followed by the final report which is this paper.

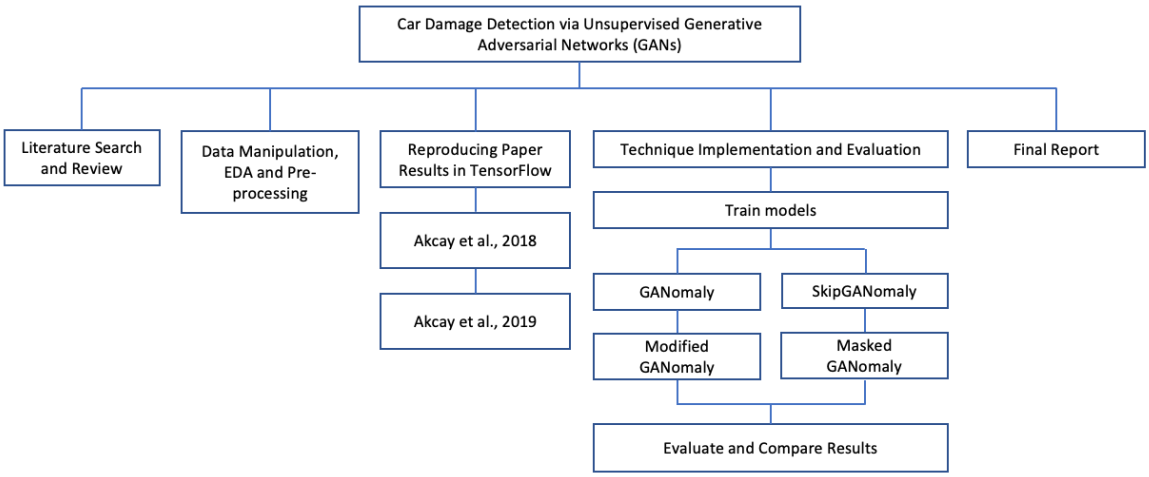


Figure 1.1. Project work plan.

1.5 Report Structure

Chapter 1 provides a brief introduction to the problem background and highlights the main objectives this project is aiming to accomplish.

Chapter 2 provides the context surrounding the theory behind GAN frameworks, and the literature review of the unsupervised GAN methods for anomaly detection which sets the background of the methods and techniques that were used throughout this research and the basis for our model selection and implementation choices.

Chapter 3 provides the detailed methods and experiments carried out to complete the objectives. The chapter will start with a brief introduction to our dataset and the pre-processing steps that are required

[4] GitHub is an internet hosting services widely utilized by Data Science community for version control and collaborative work.
[5] PyTorch is an open-source software library for machine learning and artificial intelligence framework based on Torch library originally developed by Facebook.
[6] TensorFlow like PyTorch is an open-source machine learning framework originally developed by Google.
[7] Keras is an interface for the TensorFlow library designed for deep neural network implementations.

for a robust data pipeline. The chapter will proceed with the implementation of two methods chosen for this research, GANomaly (Akcay et al., 2018) and SkipGANomaly (Akçay et al., 2019) and their model adaptations.

Chapter 4 provides an analysis of the results produced through our methods, highlighting some of the more important points and drawbacks.

In Chapter 5 we analyse our findings in light of our objectives and the broader context provided by our review of related theoretical and applied works, as discussed in Chapter 2.

Chapter 6 evaluates the project as a whole, along with suggestions for future work and improvements.

2 CONTEXT

This chapter describes the impetus for the project and the similar work done in reference literature and published scientific literature involving unsupervised anomaly detection using Generative Adversarial Networks (GANs) and sets the background of the methods and techniques that were used throughout this research. The critical context and literature review for this research have been divided into three sections. The first section explores the background of GANs and their fundamentals which are essential to have a grasp on for deeper familiarity with the subject. The second section will explore the common problems associated with the GAN framework and what to look for during the implementation stage of the project. The last section will delve into the specific models and frameworks available in the field of anomaly detection using GANs which influenced our model selection and implementation choices in the research.

2.1 Generative Adversarial Networks (GANs)

Typical machine learning problems generally include a *supervised learning* process which involves a training dataset comprised of multiple samples with input variables X and output class labels y. In this paradigm, the goal is to learn the mapping from X to y. In an *unsupervised or descriptive learning* process, however, the objective is to uncover the "interesting patterns" in the data given that only the input variables X are provided (Murphy, 2012). *Generative modelling* is an unsupervised and semi-supervised machine learning technique that learns the general patterns in input data to reproduce new instances and examples that are plausibly similar or analogous to the original dataset. Two modern examples of deep-learning-based generative modelling algorithms include Variational Autoencoder (VAE) and Generative Adversarial Networks (GANs), both of which incorporate automated discovery and learning of the general patterns in the input data.

2.1.1 Original GAN method

First proposed by Ian Goodfellow and his colleagues (Goodfellow et al., 2014), Generative Adversarial Network, or GAN for short, are a generative modelling framework that uses two sub-models trained concurrently via an adversarial process which is based on a game theoretic scenario. Goodfellow et al. (2014) proposed two Artificial Neural Network models (ANN) composed of a generator network G and its adversary, the discriminator network D, for estimating generative models via a zero-sum game, in which each is trained to fool the other until they reach the Nash equilibrium[8]. The core idea of the

[8] The Nash equilibrium is a game theory concept where no player has any incentive to unilaterally deviate from their initial strategy based on the knowledge of the other player's strategies and where each player attempts to select a strategy that maximises its own utility function

framework is, given an n-dimensional noise vector z as input (randomly drawn from a Gaussian distribution[9]), to train the generator model G to capture the data distribution $p_G(x)$ in the multidimensional vector space corresponding to points in the problem domain. This multidimensional vector space is referred to as a latent space, which is comprised of latent variables that are not directly observable by the network (Goodfellow et al., 2016). Subsequently, given the random noise z, the generator learns to output new, synthetic instances of true data distribution $p_G(x)$ that can pass for real data which subsequently become inputs for the discriminator D. Conversely, the discriminator D model's role is to distinguish between real input data x and generated $G(z)$ content, thus helping the generator learn through each iteration. In summary, the G network tries to capture the compressed representation of the data distribution, and D estimates the probability that a sample came from the training data distribution $p_{data}(x)$ rather than the generator's captured distribution $p_G(x)$. In this paradigm, zero-sum implies that anytime the discriminator accurately distinguishes real from generated samples, it is rewarded with no updates to its model parameters, whilst the generator is penalised with large updates to its model parameters. Alternatively, when the generator network starts to generate samples genuine enough to fool the discriminator, it in turn is rewarded whereas the discriminator network is penalized instead with large updates to its model parameters.

Figure 2.1 is a graphical representation of the GAN framework where both the generator and the discriminator are two neural networks trained simultaneously. Here, the generator generates a batch of samples which is later fed to the discriminator as negative training examples. The discriminator then learns to distinguish the generator's fake data from real data in the form of a classification problem, which in turn provides a signal to the generator through backpropagation in order to update its weights.

With each subsequent iteration, the weights of both the discriminator and the generator are updated in order to improve the discriminator's ability to effectively differentiate real and fake samples (minimizing J_D cost function), and the generator's ability to yield more realistic examples (minimizing J_G cost function). Over time, as the adversarial game continues, the generator learns to confuse the discriminator better by making the generated samples more realistic or close to the true distribution, while the discriminator begins to get very good at telling the generated samples from real ones. In theory, when the model reaches convergence, the generator will have captured the true data distribution $p_{data}(x)$, producing almost perfect replicas of the input data. Intuitively, this also means that the discriminator is not able to tell the difference between two sets of samples, as they are indistinguishable, where $D(x) = D(G(z)) = \frac{1}{2}$. However, reaching this convergence is an idealized case, and the model

[9] Gaussian distribution (or normal distribution) is a bell-shaped graph representing the continuous probability distribution of many random variables with mean μ and standard deviation σ.

does not need to reach this equilibrium in order to achieve a useful generator model. The objective of this mini-max game is formulated through the following mathematical formula:

$$(G, D) = min_G max_D V(D, G) = E_{x \sim p_{data}(x)}[logD(x)] + E_{z \sim p_g(z)}\left[log\left(1\text{-}D\big(G(z)\big)\right)\right] \quad (2.1)$$

where:

- V is the output value of the loss function
- x is real data sampled from real data distribution $p_{data}(x)$
- z is the latent variable, sampled from standard normal distribution $p_g(z)$
- E_x is the expected value over all real data instances
- E_z is the expected value over all fake instances
- $G(z)$ is the generator's output given noise z
- $D(x)$ is the discriminator's estimate of the probability that real sample x is real
- $D(G(z))$ is the discriminator's estimate of the probability that fake sample $G(z)$ is real

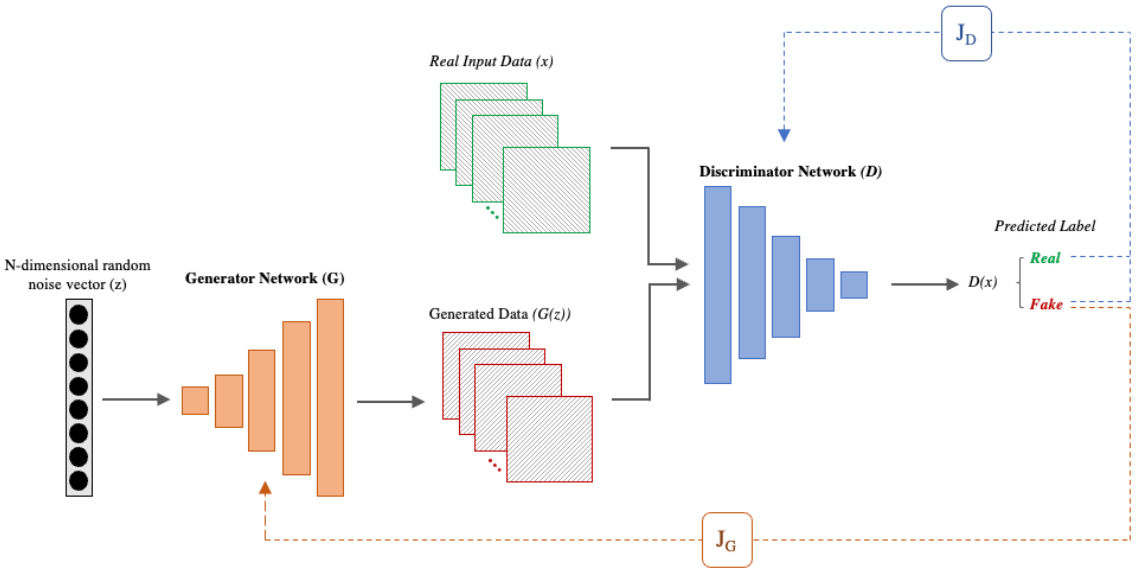


Figure 2.1. The GAN architecture with generator and discriminator sub-models trained together with objective function JG and JD respectively. The model uses an n-dimensional random noise vector (z) as input to generate new data (G(z)) which later is fed to the Discriminator network along with the original input data from the domain and classified as real or fake.

During early training, when G hasn't fully captured the compressed representation of the true data distribution yet, D can easily differentiate between real and fake images, providing insufficient gradients for G to learn. Subsequently, the above minimax loss function can cause the network to get stuck in the early stages of GAN training. Hence, Goodfellow et al. (2014) propose training G to maximise $log(D(G(z))$ instead of minimising $log(1\text{-}D(G(Z))$ to avoid vanishing gradients during the early learning process. Thus, while the discriminator attempts to make $D(G(z))$ approach zero (equation 2.2), the generator strives to bring that value to one (equation 2.3).

$$J_D = max_D\, V(D) = E_{x \sim p_{data}(x)}[logD(x)] + E_{z \sim p_z(z)}\left[log\left(1\text{-}D\left(G(z)\right)\right)\right] \quad (2.2)$$

$$J_G = min_G\, V(G) = E_{z \sim p_z(z)}\left[log\left(1\text{-}D\left(G(z)\right)\right)\right] \quad (2.3)$$

In a simple zero-sum game, the sum of all player's cost functions is always zero. In the case of GAN, it means $J_G = -J_D$, where the generator minimises the log probability of the discriminator being correct.

2.1.2 Deep Convolutional Generative Adversarial Networks (DCGANs)

GANs have rapidly evolved over the past few years, becoming one of the most popular deep learning algorithms on the premise of being able to generate highly realistic content. According to Yann LeCun, director of the IBC's research at Facebook, GANs are the most interesting idea of the past ten years in the Machine Learning field (*RI Seminar*, 2016). As of today, a large body of literature exists in the field of GANs and their variations. Figure 2.2 demonstrates the rapid headway in the number of published works in the field of GANs since it was first introduced in 2014. Over the last several years, GANs have found applications in various fields (i.e., signal processing, image synthesis and editing, and speech processing) but notable in the computer vision (Cao et al., 2019). Figure 2.3 shows GANs remarkable progress and state-of-the-art results over the years for face generation of people that do not exist in real life. As shown in figure 2.3, the 2014 example (Goodfellow et al., 2014) suffered from being noisy and incoherent. Later, the introduction of a standardised approach called Deep Convolutional Generative Adversarial Networks, or DCGAN (Radford et al., 2015) led to more stable GAN models. In fact, most GANs today are at least loosely based on the DCGAN architecture (Goodfellow, 2017).

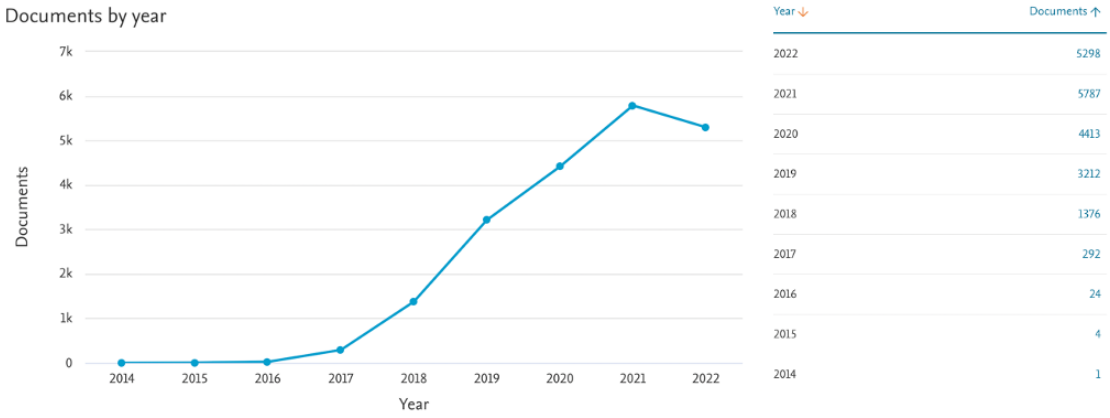


Figure 2.2. Number of publications indexed by Scopus on GANs from 2014 to 2022 ('Scopus database', 2022)

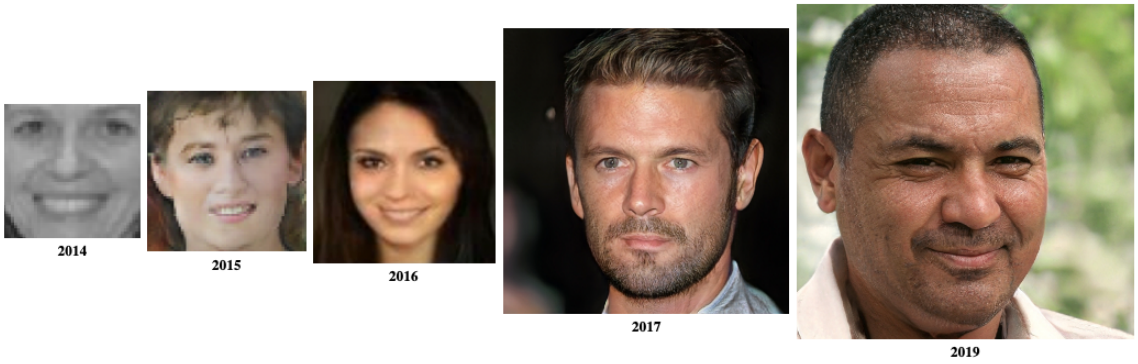


Figure 2.3. GAN progress in face generation over the years since its first introduction. 2014 (Goodfellow et al., 2014), 2015 (Radford et al., 2015), 2016 (Liu and Tuzel, 2016), 2017 (Karras et al., 2017), 2019 (Karras et al., 2019)

Fundamentals of Convolutional Neural Networks (CNNs)

CNN was first introduced as the LeNet-5 architecture, and unlike traditional feed-forward neural networks, it employs convolution operation instead of matrix multiplication, hence lowering the number of neurons required to process an image (LeCun et al., 1989). The most the framework could achieve at the time was to recognize handwritten digits. With the advancements in computing resources and accessibility to large labelled image datasets, namely ImageNet,[10] CNN was later brought back to life as the most utilized branch of the deep learning framework (Krizhevsky et al., 2012). Since then, CNN architecture has been extensively studied and perfected with time (Gu et al., 2017). In contrast to prior primitive methods with custom-tailored image filters, CNNs are able to independently learn such filters during training, with little to no pre-processing of the input data. The architecture of a typical convolutional network consists of four types of layers: convolution, pooling, activation and fully connected layer (figure 2.4).

The *convolution layer* is the first layer that extracts information from its input image using several filters carrying the main portion of the network's computational load. Each filter matrix will slide through the input matrix from the upper-left-hand side corner to the bottom-hand-side corner, each creating a feature map (Figure 2.5). Strides are used to control the number of pixel shifts over the input matrix. When the filter does not perfectly fit the image matrix, zero padding (zeros added to the end of the input sequence) or valid padding (dropping the part of the image where the filter does not fit) is used.

[10] ImageNet is a large database of more than 14 million hand-annotated images widely used in visual object detection models, leading to great advancements in computer vision and deep learning research (https://www.image-net.org/).

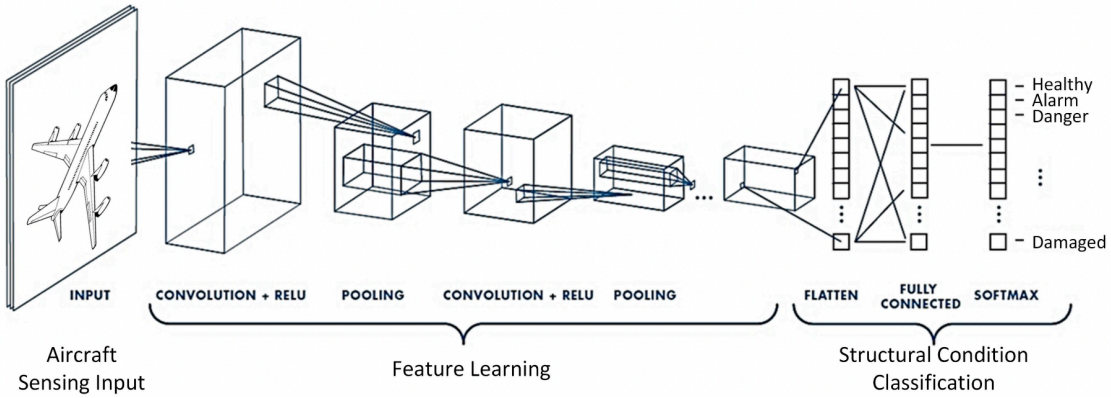


Figure 2.4. Convolutional Neural Network Architecture (CNN) with a typical configuration of convolutional, pooling, activation and fully connected layers (Tabian et al., 2019). Here the input image is fed into the first convolutional layer with a featured kernel. Following the ReLU non-linearity activation function, pooling is applied to reduce the dimensionality size. Followed by several (depending on the need) convolutional layers the network ends with flattened output that later is fed into a fully connected layer. The final classification is obtained after applying the softmax activation function which generates a normalized probability distribution in range of 0 to 1.

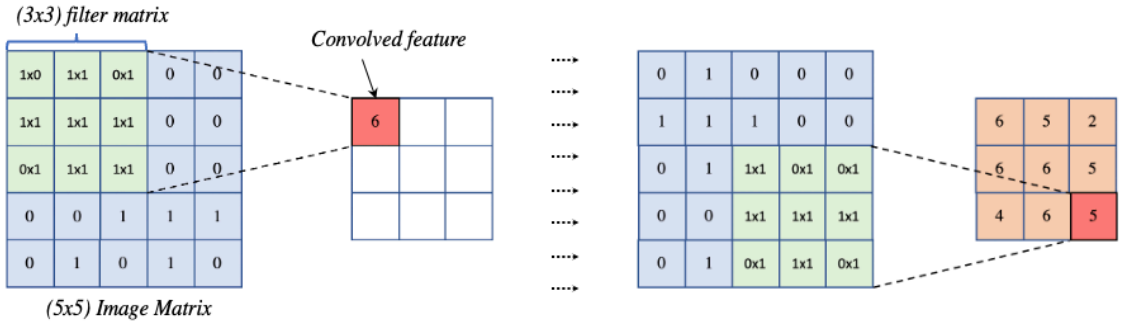


Figure 2.5. Process of convoluting the kernel with 3x3 filter matrix (kernel size) from the upper-left-hand side corner of the 5x5 image matrix to the bottom-hand-size corner outputting feature map.

Following convolutional layers, the network utilizes *activation functions* to introduce non-linearity in the output of the convolutional net. In figure 2.4 example Rectified Linear Unit (ReLU) function (Nair and Hinton, 2010) is used, where the function outputs zero for negative inputs, while the positive inputs are kept as is (equation 2.4). For positive inputs as the function increases from 0 to ∞, large variation in the input space will be translated into the outputs, thus avoiding vanishing gradients[11].

$$\text{ReLU}\,(X) = \begin{cases} 0 & x<0 \\ x & x\geq 0 \end{cases} = max(0, x) \tag{2.4}$$

The *pooling layer* is where the size of the image is gradually reduced only keeping the most important information to be passed into the next convolutional layer. Max Pooling is where the

[11] The gradients during backpropagation are very small or close to zero where little to no training can take place. Certain activation functions (i.e., sigmoid function) large variations in input is squished into a small input space, causing small derivates.

maximum value is taken from the rectified feature map in each group of *n* pixels, while Average Pooling is where average values are retained. The network ends with *fully-connected layers* where the input matrix is flattened into a one-dimensional vector space before being fed to the fully-connected layer, followed by activation functions such as softmax (outputs normalized probability distribution proportional to the exponentials of the input in the range of 0 to 1, all the elements adding up to 1) or sigmoid (outputs probability distribution of *N* possible outcomes in the range of 0 to 1 or -1 to 1) for multi-class classification.

DCGAN architecture

The authors of the original DCGAN paper (Radford et al., 2015) were able to identify a CNN architecture that was best suited for GAN models. Three main modifications to the typical CNN architecture were performed, resulting in higher-resolution images and more stable training.

As discussed in the previous section, modern convolutional neural networks (CNNs) use alternating convolution and pooling layers followed by one or more fully-connected layers. The authors of the original DCGAN paper (Radford et al., 2015) were able to modify several components of the CNN pipeline resulting in more stable GAN training and higher-resolution images. DCGAN framework utilizes the all-convolutional net (Springenberg et al., 2015) in which the modified CNN architecture uses convolutional layers with a stride of two for dimensionality reduction instead of max-pooling layers (Figure 2.6). This simple modification allows the network to learn its own spatial upsampling (generator) and downsampling (discriminator).

In DCGAN network the first layer of the generator G takes a uniform noise distribution *z* as input, reshaping it into a four-dimensional tensor before feeding it into a convolutional stack. For the discriminator *D*, the last convolution layer is flattened and then fed into a single sigmoid output. Additionally, both models omit the fully-connected layers on top of convolutional features (Figure 2.6).

The third modification to the standard CNN architecture is the use of *Batch Normalization* where each input is normalised to have zero mean and unit variance for each training mini-batch (Ioffe and Szegedy, 2015). Batch Normalization helps remove internal covariate shift[12] and that way stabilising the learning process. Additional to making it possible to use higher learning rates, it also acts as a regularizer which makes the need for Dropout[13] layers redundant.

Except for the last layer, which utilises the *Tanh* function (hyperbolic tangent of the input), the generator uses *ReLU* activation (Nair and Hinton, 2010). Different to the original GAN paper

[12] The phenomenon where the model is hard to train with saturating nonlinearities, by the fact that the distribution of each layer's input changes during training, which in turn slows down the training requiring lower learning rates and careful parameter initialization (Ioffe and Szegedy, 2015).
[13] A regularization technique used to prevent overfitting, by randomly dropping out nodes in the neural network and setting them to zero.

(Goodfellow et al., 2014) however, Discriminator employs the *Leaky Rectified Linear Unit* (LeakyReLU) activation function (Maas, Hannun and Ng, 2013) (Xu et al., 2015) instead, which has been shown to work better for higher resolution modelling. LeakyReLU is a variant of ReLU, but instead of making all negative values 0, a LeakyReLU allows small non-zero values to pass through with constant gradient α, in order to prevent the "dying ReLU"[14] problem. As a result, it also makes the gradients from the discriminator flow stronger into the generator.

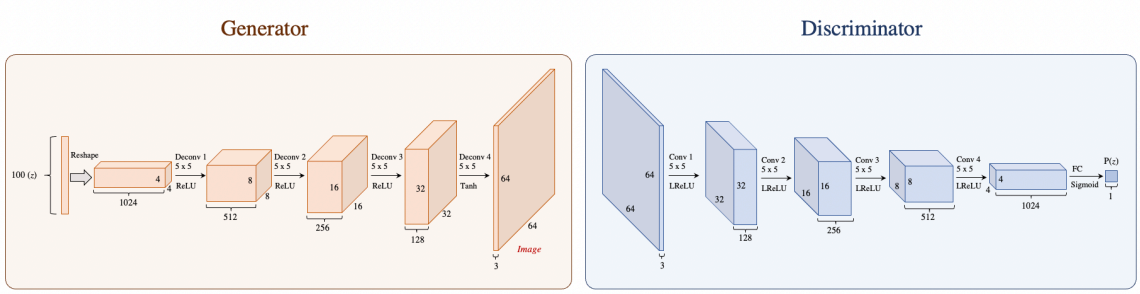


Figure 2.6. DCGAN architecture with generator and discriminator networks. Generator (G) takes a 100-dimensional uniform distribution z and converts the high-level representations acquired through transposed convolutional layers with kernel size 5x5 and strides of (2, 2) to generate 64x64x3 pixel images. The G network uses the ReLU activation function except for the last layer where Tanh is utilized. The discriminator (D) network then takes the generated image and outputs a probability distribution classifying the image as real or fake. The D network utilizes convolutional layers with a kernel size of 5x5 and strides of (2, 2) and LeakyReLU activation function. In contrast to the original CNN architecture, no pooling and fully-connected layers are utilized in the structure.

2.2 Potential problems with training GANs

Despite its superior ability to generate real instances of the input data, and the considerable attention it has received over the years with some state-of-art results, it is still important to touch upon some of the weaknesses and problems associated with training a GAN model. Mainly due to their instability during training and their sensitivity to hyperparameters, GANs are notoriously difficult to train and adapt to different datasets. This section will cover some fundamental problems that many GAN models suffer from, making it one of the elusive models to train. The following section will define the major drawbacks and complexities associated with training a GAN and several methods proposed in recent literature for mitigating them.

[14] When most inputs to the ReLU neurons are in the negative range, and the outputs return zero, a significant portion of the network becomes inactive (weights do not update) due to weak gradient flow during backpropagation.

2.2.1 Diminishing gradients

When the discriminator is too confident in its predictions between real and fake images, the generator training can fail due to vanishing gradients. In effect, the gradients that the generator receives during backpropagation are vanishingly small for it to make enough progress in its learning (Arjovsky and Bottou, 2017). As discussed in section 2.1.1, Goodfellow et al. (2014) proposed modifying the generator loss so that the generator tries to maximize log *D(G(z))* in order to provide stronger gradients for the generator and deal with the vanishing gradients problem. One other widely used solution is to use the *Wasserstein loss* (Arjovsky et al., 2017) which is designed to prevent diminishing gradients despite how optimally we train our discriminator.

2.2.2 Mode Collapse

Mode collapse is one of the most common failure cases while training GAN models. Mode collapse, also known as the Helvetica scenario, is a problem that occurs when the generator learns to generate samples with a limited diversity for different latent spaces (Goodfellow, 2017). In simplified terms, the generator produces a small set of outputs that it finds to be sufficient to easily trick the discriminator, which is its main objective. It will generate similar data outputs under the assumption that the goal of fooling the discriminator is achieved, that way over-optimizing the whole system to that single type of output. The best way to detect a mode collapse is by looking at the generated images. When the training data is too big, we can identify mode collapse by directly observing the loss graphs. In such cases, if we observe the oscillating generator loss over time while discriminator losses for fake and real images hover around the same value, this could be an indication of a mode collapse (Figure 2.7). If both generator and discriminator losses drop down to zero in the early epochs, that may also be a problem, indicating the generator has found a set of fake images that are easy for the discriminator to identify.

Research in recent years has proposed several remedies to force the generator to increase its diversity and coverage of the data distribution and consequently prevent the generator from optimising to a single fixed discriminator. One of the solutions to avoid such undesirable local equilibria is to use the *Wasserstein loss* (Arjovsky et al., 2017). This loss function enables the discriminator to reject the similar outputs that the generator produces, forcing the generator to generate a different set of samples every time. *Unrolled GANs* (Metz et al., 2017) is another method that can be used, forcing the generator loss to incorporate both the current discriminator's classifications and the subsequent discriminator iterations. Additionally, *Minibatch Discrimination* has also been shown to prevent the generator from collapsing to a parameter setting where the similarities of generated images increase (Salimans et al., 2016). During Minibatch Discrimination, real and generated images are fed to the discriminator separately in different batches, allowing the discriminator to look at multiple data examples in combination. Using this scope, the discriminator will detect fake images and penalise the generator if

the mode collapses. *Multi-Scale Gradients for GANs* (MSG-GAN) is another technique used for addressing mode collapse, which allows gradients to flow from the discriminator to the generator at multiple scales (Karnewar and Wang, 2020).

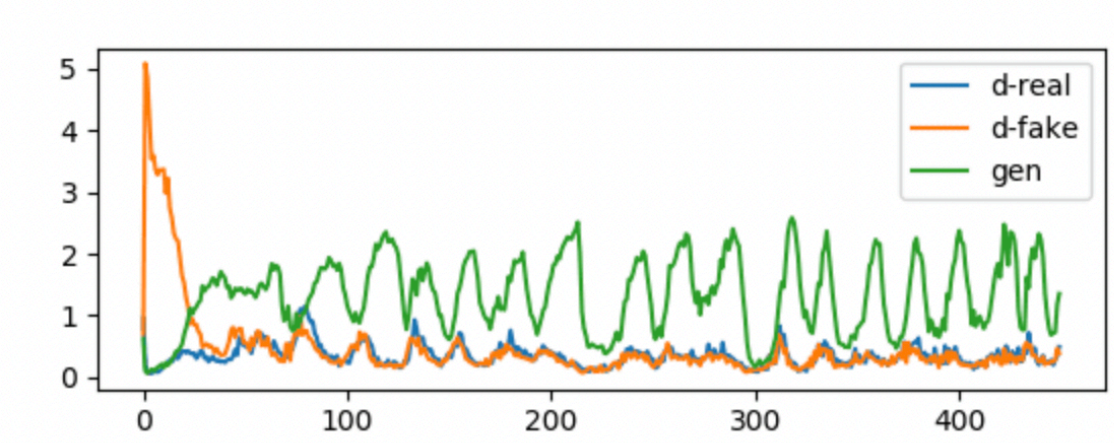


Figure 2.7. An example of a loss graph for a GAN model with a mode collapse (Brownlee, 2019). In this plot, we can observe the generator loss (green) oscillating from high values to low values approximately over 25 model updates (batches). Similar oscillations but at a smaller scale are seen in the discriminator loss for real (orange) and fake samples (blue).

2.2.3 Non-convergence

GAN non-convergence typically refers to a failure in finding a Nash equilibrium between the discriminator and the generator networks. One way to identify this type of failure is through a loss graph, where discriminator loss approaches zero, while most times, generator loss may also increase and continue so over time. In layman's terms, the generator starts to output low-quality images that the discriminator can easily identify as fake images. This behaviour is usually associated with an aggressive *Adam optimisation algorithm* (Kingma and Ba, 2017), or with either very large or very small kernel sizes in the models. One of the most effective ways to combat the non-convergence problem is to make the training of the discriminator network more difficult by adding noise to the discriminator's inputs and increasing the complexity of the discriminator training. This way, we are able to give some stability to the data distributions of the two competing networks. Another simple solution is to add one-sided label smoothing to the discriminator (Salimans et al., 2016). Here, instead of labelling the real images as 1, we can set it to a lower value like 0.9, leaving negative labels set to 0. This, in turn, will prevent our discriminator from being overconfident in its classification and relying on a limited set of features. Additionally, Roth *et al*. (2017) introduced a regularisation scheme where by adding a penalty to the weighted gradient-norm of the discriminator, we can prevent non-convergence and increase stability.

2.3 Anomaly Detection using GANs.

This section focuses on the research progress and application of GANs in anomaly detection. One of the critical problems that researchers in the Machine learning (ML) and Artificial Intelligence (AI)

fields are trying to solve is anomaly detection. It is a known fact that the anomalies in data can translate to significant, if not critical, information in a wide variety of application domains. Anomaly detection is the task of identifying abnormal patterns in the data that deviate or differ significantly from the majority of instances and do not conform to a well-defined notion of normal behaviour (Chandola et al., 2009). Several deep-learning methods, specifically generative models due to their ability to perform distribution fitting, have shown remarkable progress in the field of solving the anomaly detection problem. Variational AutoEncoder (VAE) and GAN are the two most representative frameworks of the generative models that have become one of the best methods for anomaly detection. Even though GANs are known to generate higher quality results than VAEs, VAEs suffer less from mode collapse problems (Hong et al., 2019).

There is a large volume of studies proposing anomaly detection using GANs, however, due to space limitations, we are unable to explore it all in detail. Xia et al., (2022) have gathered an extensive survey in that direction, which includes a detailed commentary of the aforementioned works and references therein.

Widely regarded as a seminal work, AnoGAN (Schlegl et al., 2017) is the first model in the field of anomaly detection using adversarial training. The model utilizes DCGAN architecture training only on healthy data (with no anomalies), with the purpose of learning the mapping from latent space representation z to normal (non-anomalous) images x. This learned representation is later used during inference to map unseen samples back to the learned feature representation, and when an anomalous image is introduced, it reconstructs a non-anomalous image. The difference between the reconstructed and original image is used to map the anomalous region. However, this method represented some drawbacks in the form of huge performance issues due to the enormous computational complexity of remapping back to latent vector space. The performance issue arises due to the fact that AnoGAN requires an optimization process during inference to find the latent space vector to reconstruct an image analogous to the generative distribution.

Efficient GAN-Based Anomaly Detection (EGBAD) was later introduced by Zenati et al., (2019) which uses a Bidirectional Generative Adversarial Networks (BiGAN) architecture (Donahue et al., 2017), to address the performance issues put forward by the initial AnoGAN method. Unlike AnoGAN, EGBAD was able to compute anomaly scores without optimizations during the inference, that way substantially improving test-time performance.

To address the performance issues the authors of the original AnoGAN method later introduced Fast Unsupervised Anomaly Detection with Generative Adversarial Networks (f-AnoGAN), where the inverse mapping from image space to latent space was removed.

Going forwards, this research will focus on GANs in outlier detection by studying the two of the recent architectures proposed in the literature, namely GANomaly (Akcay et al., 2018) and Skip-GANomaly (Akçay et al., 2019).

2.3.1 GANomaly

Motivated by both AnoGAN and EGBAD, the authors were able to come up with a framework where by only training on normal samples the generative network learns the latent region of the input space producing state-of-art results in the field of anomaly detection. GANomaly differs from its contemporaries in that it uses encoder-decoder-encoder sub-networks in the generator, which maps the input image to a lower dimensional latent vector, which is then used to reconstruct an image resembling the data distribution of the non-anomalous samples.

GANomaly employs both the DCGAN architecture and Autoencoders. Autoencoders are neural networks that aim to replicate their inputs as outputs by compressing the input into a latent-space representation and generating an output from this representation. An autoencoder consists of two primary components: an encoder and a decoder. The encoder's job is to learn to compress the input data into an encoded latent space representation. This latent region becomes the bottleneck of the whole network. The decoder network then takes this encoded representation and aims to recreate the original data from it pushing the generated data to be as close to the original input as possible.

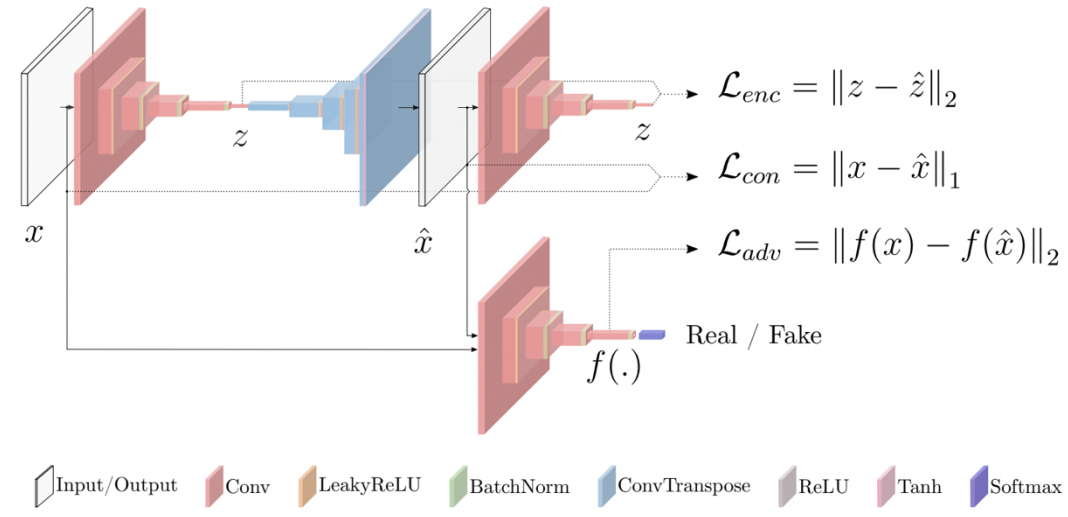


Figure 2.8. The pipeline of the proposed GANomaly framework for anomaly detections. The generator consists of two sub-networks. The first subnetwork is a bow tie autoencoder network, which is responsible for mapping the input to latent space and remapping it back to input data space (reconstruction). The second sub-network is a single encoder which maps the latent space representation of the reconstructed image, which is later compared with the input image's encoded latent space. The model employs a custom objective function using three different losses (Adversarial, Contextual and Encoder losses). (Akcay et al., 2018)

The Generator network consists of two sub-networks: a bow tie autoencoder network and a single encoder network. The autoencoder sub-network is responsible for the reconstruction of the input image using the learned latent space representation (bottleneck of the network). A single stand-alone encoder network on the other hand is responsible for mapping the latent space of the reconstructed data space. The Generator objective function is combing three loss function:

Adversarial loss – feature matching loss for adversarial learning

Contextual loss – L_1[15] distance between the input and the reconstructed image

Encoder loss – L_2[16] distance between the latent vectors of the input image and the generated image

The Discriminator network is similar to the single encoder sub-network, except it ends with a sigmoid function and returns real/fake classification. An aggressive modifier is generally bad news for GANs, hence, the GANomaly model rightfully utilizes Adam optimizer (Kingma and Ba, 2017). Adam optimiser has two hyperparameters (β_1, β_2) which are used for calculating the running average of the gradients and their square. From the initial to later stages of training the β value increases along with the aggressiveness of the optimizer.

2.3.2 SkipGANomaly

The authors of the original GANomaly method introduced an improved version of the method which then they called SkipGANomaly (Akçay et al., 2019). Similar to the GANomaly framework it employs DCGAN architecture and Autoencoders. But the authors went a step further and incorporated *skip connections* between the encoder and the decoder components of the Generator network. The concept was first introduced by Ronneberger et al., (2015) as U-Nets framework, which has an encoder and a decoder part. The layers in the encoder component are skip-connected and concatenated with the corresponding pair in the decoder network. This allows the framework to use the fine-grained details learned during the mapping of the latent space to reconstruct an image in the decoder part. Akçay et al., (2019) were able to show that by applying skip connections in the network the framework is able to capture the multi-scale distribution of the non-anomalous data in high-dimensional image space. The idea behind incorporating skip connection in the autoencoder architecture allowing the information to bypass the bottleneck is to improve the sharpness of the reconstructed images.

All other aspects of the framework are the same as in the original GANomaly method, apart from the adversarial loss function which applies the original adversarial loss proposed by Goodfellow et al., (2014) (equation 2.3). The objective of the overall framework is to minimise the reconstruction loss

[15] L_1 distance, also known as Manhattan distance, is the sum of the magnitudes of the vectors in a space, or in other words the sum of absolute difference of vector components.
[16] L_2 distance, also known as Euclidean norm, is the shortest distance between two points.

within both the image and vector spaces during training and learn the true distribution of non-anomalous data space. By learning the distribution of the non-anomalous domain, the authors were able to detect the abnormality based on the deviations from the both high-dimensional image space and lower-dimensional latent vector space encoding.

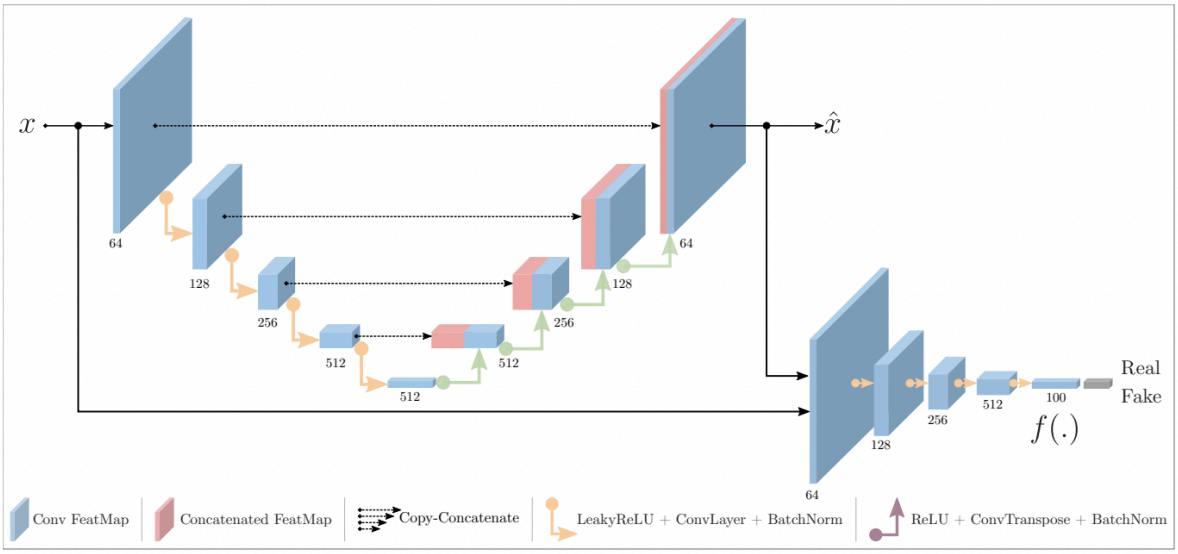


Figure 2.9. Pipeline of the proposed SkipGANomaly framework for anomaly detections. The generator consists of a single bow tie autoencoder network, which is responsible for mapping the input to latent space and remapping it back to input data space (reconstruction). The discriminator is responsible for real/fake classification as well as generating the latent vectors to be later used in the encoder loss function. (Akçay et al., 2019)

3 METHODS

The chapter will start with a brief introduction about the dataset used in the project with the analysis and pre-processing steps necessary to conduct this research. Henceforward, we are going to implement various versions of unsupervised anomaly detection via adversarial training, namely GANomaly and SkipGANomaly, methods originally proposed by Akcay et al. (2018, 2019).

3.1 Dataset

The dataset used in this project is a proprietary dataset that Tractable AI acquired in 2019 from a third-party software development company (NuGen IT[17]). The Nugen dataset contains ~9 million 'jpeg' images obtained from historic auto collision claims. The images are categorised by claim number, each containing multiple photographs of the same vehicle from different vantage points (usually taken by the repair shop or the owner). Each image is given a one-hot-encoded label (1 for True, 0 for False), based on the parts present in each image and whether or not they are damaged.

Similar to many real-world datasets, the Nugen dataset proved to be complex, necessitating extensive analysis and pre-processing. Initial Exploratory Data Analysis (EDA) revealed that some of the images labelled as undamaged did in fact contain damages from prior incidents that were not reported under the current collision claim. Due to the rarity of these artefacts in our dataset, we anticipate our generative model to learn to ignore these outliers through training.

Numerous categories can be found in the Nugen dataset. These categories include the vehicle's manufacturer, model, body type, colour, degree of damage, and so on. In order to guarantee a better generalization, we had to narrow down our problem domain to a single make and model, namely Toyota Camry, which is the most common vehicle type in our dataset (602,374 image samples). We further filtered our images to only include Camrys manufactured between 2007 and 2011, as their exterior designs are similar by default, which further trimmed our data samples to 444,826 images. Lastly, we chose to only use the silver colour Camrys, as GAN models often show bias for colour hues, being susceptible to that feature. By pruning our dataset to a smaller domain, we enable our generative models to better map the input images to a representative lower dimensional space that can be generalized more easily across all data samples.

Upon further investigation, we discovered that some of the images contained interior and concealed vehicle components, which we were required to remove from our dataset as we only wish to include

[17] NuGen IT is a software development company that provides streamlined claims process for insurance companies, providing more efficient parts and repair ecosystem. https://go.oeconnection.com/nugenit

images with exterior parts. Despite our best efforts, some of the images with an open boot/bonnet or car doors still managed to leak through the filtering process, introducing additional noise/outliers in our dataset. As depicted in Figure 3.1, the true nature of the final training dataset of undamaged cars contains undesirable samples; however, because Tractable AI did not have an explicit dataset of undamaged cars, we were forced to work with what we had available. In the long run, it proved to be beneficial to test our frameworks' robustness to noise represented as damages, open exterior parts showing the interior of the cars, low-quality images, and foreign objects in view.

Figure 3.1. Random samples extracted from the final training set of 'undamaged' car images, containing artefacts (marked in red boxes) such as damages, low-quality images, open exterior parts, and foreign objects (i.e., measuring tape, humans, data stamps, etc.) in view.

There is a total of 23 distinct labels of exterior parts representing our panels in scope:

["fbumper", "bonnet", "flwing", "frwing", "fldoor", "frdoor", "rldoor", "rrdoor", "blwing", "brwing", "bootlid", "bbumper", "grille", "lheadlamp","rheadlamp", "lmirror", "rmirror", "ltaillamp", "rtaillamp"]

The final dataset consists of photographs with at least three exterior car components in view and was split into damaged and undamaged subsets based on the available one-hot-encoded damage labels. After filtering for corrupt .jpeg data, we were left with 6161 samples of undamaged cars and 10227 damaged cars.

The original raw data is stored in an S3 Object Storage solution offered by Amazon Web Services[18] (AWS). Given that we are training our models on GPUs[19], speeding up our data pipeline was very paramount. For that purpose, we utilize `tf.data` which is a TensorFlow API that provides all the necessary building blocks to assemble our data, such as processing filenames, loading image bytes,

[18] Amazon Web Services (AWS) is one of the most popular cloud computing platforms available today (https://aws.amazon.com/)

[19] Graphics processing unit (GPU) is a specialized processor designed to accelerate model training by parallel computing capable of processing many pieces of data simultaneously.

decoding jpeg files, and resizing, augmenting our images and batching. As a means of keeping our computational costs low while preserving the visibility of small anomalies such as dents, scratches, etc., we resize all input images to 256x256x3 (*height* x *width* x *channels*) dimensions. As our training data is relatively small, we perform various augmentation methods (i.e., horizontal flip, zoom) on our data to increase its size and diversity. Additionally, use of `.prefetch()` function in our data pipeline allows the reading of the next batch of data while the model is executing a training step.

3.2 Original GANomaly

This section describes the first method, GANomaly, originally proposed by Akcay et al, (2018). The original architecture was built using a PyTorch library and is publicly available in a GitHub repository[20] trained on benchmark datasets. To gain a deeper familiarity with the model and more leeway for customization, we build our model from scratch but instead using TensorFlow and Keras libraries.

Figure 3.2 depicts the model architecture for Generator and the Discriminator networks in the GANomaly model. The generator consists of two encoders which downsample an image with dimensions of 256x256x3 to a 4x4x2048 matrix after which we retrieve a 1x1x100 latent vector space vector (z, $\hat{z}$). The single decoder network on the other hand, upsamples a given 1x1x100 latent vector to a generated image ($\hat{x}$) with dimensions of 256x256x3. Each Encoder network contains seven downsampling blocks consisting of (with the exceptions of the first and last downsampling blocks) a convolutional layer with a kernel size of 4x4 and strides of 2x2, batch normalisation with epsilon=1e-05 and momentum=0.1, and LeakyReLU activation layer with alpha=0.2. As explained by Ioffe and Szegedy (2015), batch normalisation works by normalising the input features of the layer to have zero mean and unit variance. Radford et al., (2015) have shown that using batch normalisation helps deeper generative models to work without falling into mode-collapse. The first downsampling block does not contain any batch normalization, while the last block omits both batch normalization and the activation function and has strides of 1x1.

The decoder network which contains seven upsampling blocks, utilises ReLU activation function except for the last layer which uses Tanh function (as well as skipping batch normalisation). The Generator network returns three outputs consisting of the first latent vector z of the input image x, generated image $\hat{x}$, and the second latent vector $\hat{z}$ of the generated image $\hat{x}$. Similar to the generator's encoder network, the discriminator network consists of seven downsampling blocks with the exception of sigmoid activation function used in the last layer. The discriminator network produces two outputs, one being the feature vector before last layer and a real/fake classification after sigmoid layer.

[20] Publicly available GitHub repository for GANomaly implementation https://github.com/samet-akcay/ganomaly

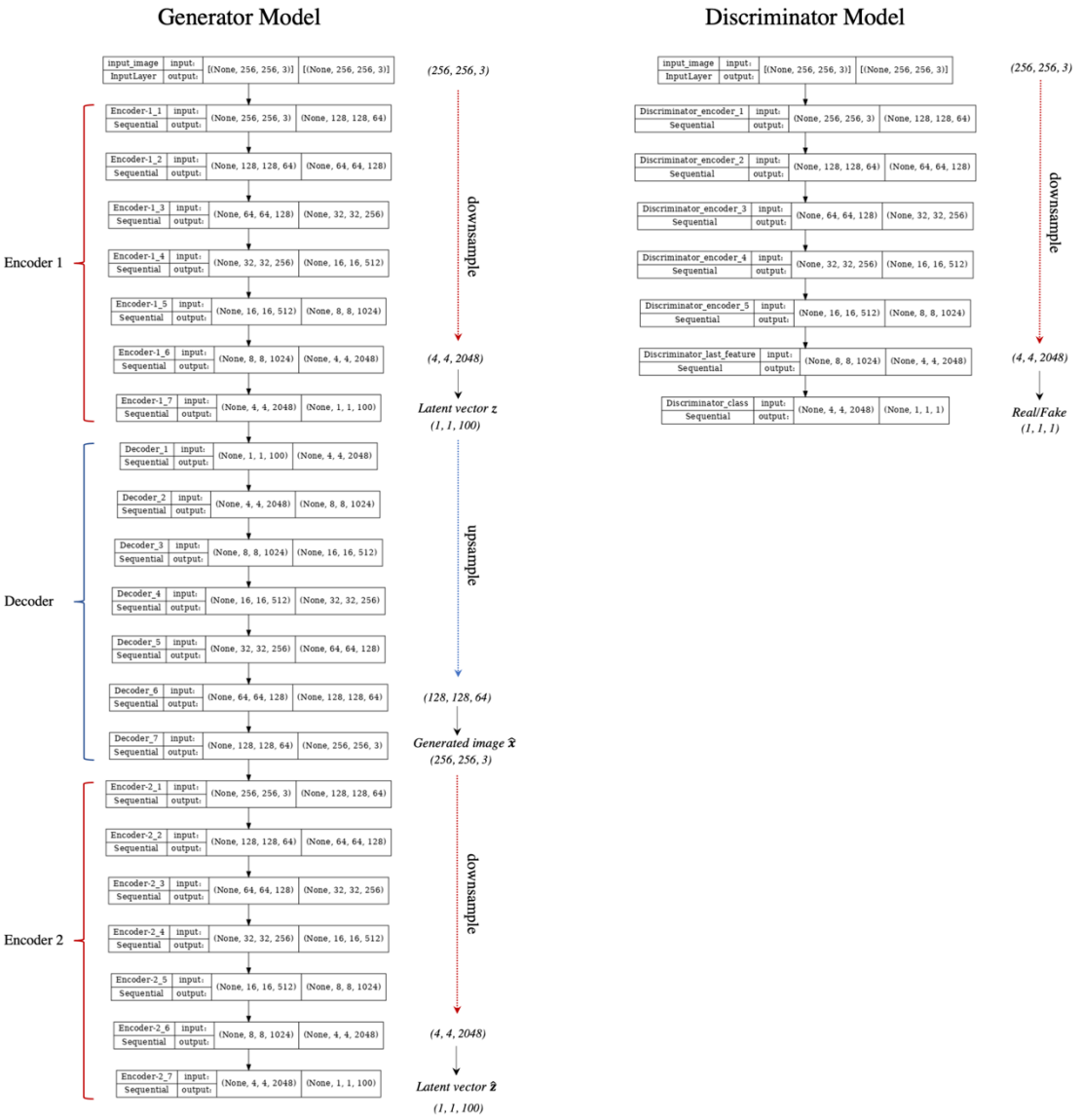


Figure 3.2. Original GANomaly architecture for generator and the discriminator networks respectively. The generator network consists of on decoder network with seven upsample blocks and two encoder networks each with seven downsampling blocks. The generator network takes an input image and returns three outputs, z, $\hat{z}$, and $\hat{x}$. The discriminator network has a similar encoder network structure with the last layer ending with a sigmoid activation function. The discriminator returns three outputs, a feature vector for real and fake images and the probability predictions in the form of classification for real and fake (0/1).

Table 3.1 demonstrates the number of trainable and non-trainable parameters for generator and discriminator models respectively.

Generator	Discriminator
Total params: 143,974,144 Trainable params: 143,950,208 Non-trainable params: 23,936	Total params: 44,747,264 Trainable params: 44,739,328 Non-trainable params: 7,936

TABLE 3.1. The number of total, trainable and non-trainable parameters for the generator and the discriminator networks respectively in the original GANomaly method.

Objective Function

Different sets of custom loss functions are used for generator and the discriminator models respectively as proposed in the original paper (Akcay et al., 2018). The loss function for the generator is the sum of three different loss functions (Adversarial loss, Contextual loss, and Encoder loss) with corresponding weighting parameters ($\omega_{adv} = 1, \omega_{con} = 50, \omega_{enc} = 1$) that are responsible for adjusting the individual losses to the overall objective function (equation 3.1).

$$\mathcal{L}_G = \omega_{adv}\mathcal{L}_{adv} + \omega_{con}\mathcal{L}_{con} + \omega_{enc}\mathcal{L}_{enc} \tag{3.1}$$

Adversarial loss uses a feature matching loss for adversarial learning, which changes the cost function for the generator to minimise the statistical difference between the feature representation from the intermediate layer of the discriminator (equation 3.2). Feature matching uses maximin mean discrepancy to train generator networks (Dziugaite et al., 2015). Adversarial loss is computed by finding the L_2 distance between the feature representation of the original and the generated images. Since features are computed per minibatch, which varies with each batch, this randomness introduces additional difficulty for the discriminator, making it harder to overfit itself, reducing the likelihood of any mode collapse.

$$\mathcal{L}_{adv} = \mathbb{E}_{x\sim p_X}\left\| f(x) - \mathbb{E}_{x\sim p_X} f(G(x)) \right\|_2 \tag{3.2}$$

Additional to adversarial loss, in order to optimize the generator towards learning contextual information about the input data a *contextual loss* function is also introduced to the generator's overall objective function (equation 3.3). Contextual loss measures the L_1 distance between the original (x) and the generated images ($\hat{x}$). It has been shown that the use of L_1 distances yields less blurry results than L_2 distance (Isola et al., 2018).

$$\mathcal{L}_{con} = \mathbb{E}_{x\sim p_X}|x - G(x)|_1 \tag{3.3}$$

Last but not least, in addition to the two losses introduced above, *encoder loss* is used to minimize the distance between the latent vectors, allowing the generator to learn to encode features of the generated images for original samples (equation 3.4). Encoder loss is computed using the L_2 distance between the encoded latent features of the input image (z) and the generated image ($\hat{z}$).

$$\mathcal{L}_{enc} = \mathbb{E}_{x\sim p_X}\|G_E(x) - E(G(x))\|_2 \tag{3.4}$$

The idea behind using all three losses in the overall objective function is to enforce the Generator to produce images that are not only contextually sound and optimised towards normal samples, but also reduce the instability of the overall GAN training.

The loss function for the discriminator follows the original adversarial loss function proposed by Goodfellow et al., (2014) (equation 3.5).

$$\mathcal{L}_D = \mathbb{E}_{x \sim p_X}\left[log\left(D(x)\right)\right] + \mathbb{E}_{x \sim p_X}\left[log\left(1 - D\left(G(x)\right)\right)\right] \quad (3.5)$$

Both generator and discriminator models use Adam optimiser with learning rate $\alpha = 0.0002$, and hyper-parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$ which are initial decay rates used in estimating first and second moments of the gradients. Both β_1 and β_2 are exponentially multiplied by themselves at the end of each training step (batch).

3.3 Modified GANomaly

After initial inspections of the results from the first method using the GANomaly model with parameters similar to the ones provided in the original paper (Akcay et al., 2018), further improvements were applied to improve the model performance.

Larger kernel size: In this second experiment, we choose to increase the "kernel size" to 5x5, as large kernels cover more data by probing a greater number of pixels in the previous layers and therefore looking at more information. Smoothness is preserved when the kernel size at the top of the convolutional layers is big enough. Moreover, as later explained in section 4.2, a smaller kernel size in the discriminator can lead the discriminator to rapidly approach zero.

Large latent dimension: Initial experiment had a latent dimension of 100, which we believe might have further constricted our model's ability to accurately reconstruct the training data further. By having a larger latent dimension, we make sure the generator preserves more information from the encoder to be later passed into the decoder for reconstruction.

Soft and noisy labels: Discriminators in GAN networks often suffer from overconfidence, since the number of features it uses to classify an object is constricted, which the generator can exploit in order to fool the discriminator. This in turn forces the optimisation to turn too greedy producing no long-term benefits. This will become clearer in the following chapter on the findings when the results of the first experiment show that having hard labels (fake-0, real-1) kills all learning process early on where the discriminator approaches zero loss very rapidly. Soft labels are applied in the discriminator alone (this is not required when training the generator) where random numbers between 0 and 0.1 are generated to represent 'zero' labels (fake) and random numbers between 0.9 and 1 are generated to represent 'one' labels (real). Additionally, we add some noise to training labels where 5% of the images that were fed to the discriminator, where the labels are randomly flipped for generated and real images (real images labelled as fake and fake images labelled as real). This makes the discriminator's job a lot more difficult and prevents it from making overconfident decisions.

Adding noise to the discriminator: Another simple improvement that can be applied in order to make the training of the discriminator more difficult making it beneficial for the overall stability is to

add Gaussian noise at the very beginning of the discriminator. This will add additional restraint on the discriminator that will prevent it from learning trivial transformations right away.

Two Time-scale Update Rule (TTUR): The procedure entails setting different learning rates for the two models (generator and discriminator). Mathematical proof of convergence to Nash equilibrium was provided in the paper by the authors who first introduced the method, and they demonstrated that state-of-the-art results could be obtained by implementing DCGAN with varying learning rates (Heusel et al., 2018). The idea behind the method is to force the generator to make smaller steps to fool the discriminator, that way preventing it from choosing fast and not realistic solutions for the sole purpose of winning the zero-sum game. It is also recommended by the authors to train the discriminator with a learning rate 4 times greater than the generator, hence the learning rates we chose are 0.0008, 0.0002 respectively.

Modifying Adversarial Loss function: We decide to change the cost function for adversarial loss for a better optimisation goal. We add an additional penalty to the adversarial cost function to enforce constraints by combining equation 3.2 with 2.3. The modified Adversarial loss function for the generator is as follows:

$$\mathcal{L}_{adv} = \mathbb{E}_{x\sim p_X}\left\| f(x) - \mathbb{E}_{x\sim p_X} f(G(x)) \right\|_2 + \mathbb{E}_{x\sim p_X}\left[log\left(1\text{-}D\big(G(x)\big)\right)\right] \tag{3.6}$$

As seen in the network architecture depicted in figure 3.3, unlike in the model of the first experiment, we downsample our layers down to 4x4x512, that way reducing the number of trainable parameters substantially (more than twice) (table 3.2).

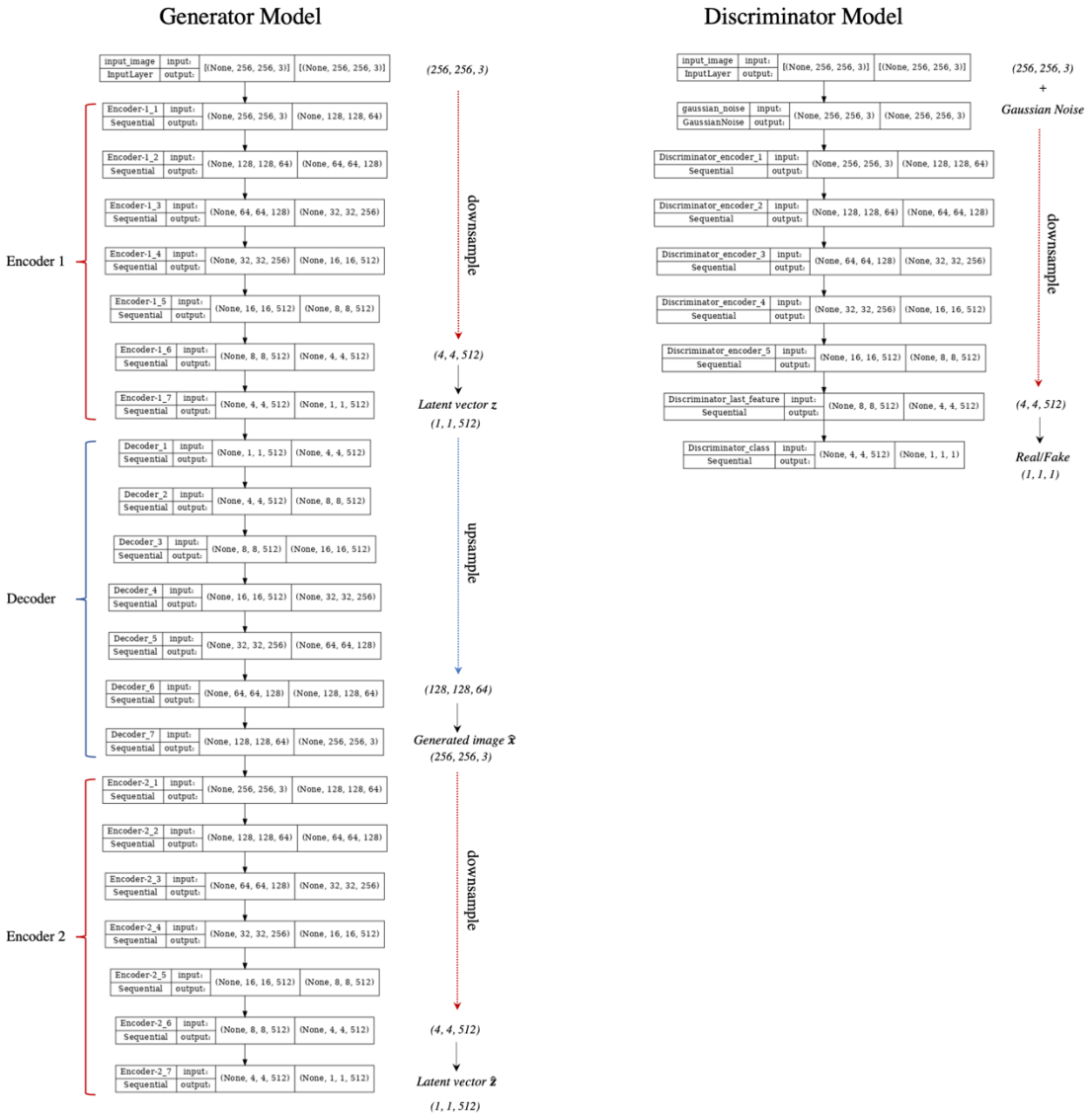


Figure 3.3. Modified GANomaly architecture for generator and the discriminator networks respectively. The network is similar to original GANomaly method (figure 3.1) with one exception being the encoder network downsampling blocks going down to 4x4x512 matrix.

Generator	Discriminator
Total params: 64,842,880 Trainable params: 64,831,232 Non-trainable params: 11,648	Total params: 17,428,672 Trainable params: 17,424,832 Non-trainable params: 3,840

TABLE 3.2. The number of total, trainable and non-trainable parameters for the generator and the discriminator networks respectively in the modified GANomaly method.

3.4 GANomaly with Spectral Normalisation

As explained by Ioffe and Szegedy (2015), batch normalisation works by normalising the input features of the layer to have zero mean and unit variance. Radford et al., (2015) have shown that using batch

normalisation helps deeper generative models to work without falling into mode-collapse. Spectral normalisation however is a different kind of regularisation method that is applied to convolutional kernels that can greatly help stabilise the training of the discriminator. Miyato et al., (2018) have experimentally confirmed that spectrally normalised GANs are capable of generating much better-quality results than their previous counterparts. In essence, spectral normalisation constrains the Lipschitz[21] constant of the convolutional filter. By controlling this constant of a network, we can improve adversarial robustness by preventing gradients from exploding or vanishing. In simple terms, spectrally normalised GANs have an upper limit on their gradient norm, which is how it mitigates exploding or vanishing gradients.

In this third experiment, we take the model from our second method and apply spectral normalisation to the discriminator's convolutional kernels only. Previous research has shown that spectral normalization slows GAN training, which can be easily mitigated by using TTUR. As in the previous experiment, we apply a higher learning rate for optimising the discriminator. A higher learning rate ensures the discriminator absorbs the larger part of the gradient signal, that way moderating the slow learning of the regularised discriminator. Everything else is kept the same, i.e., model architecture (figure 3.2), trainable parameters (table 3.2) and hyper-parameters (section 3.3).

3.5 Original SkipGANomaly

This section describes the fourth method, SkipGANomaly, which is an improved version of GANomaly, originally proposed by Akçay et al., (2019). The original architecture was built using a PyTorch library and is publicly available in a GitHub repository[22] trained on benchmark datasets.

Figure 3.4 demonstrates the model architecture for generator and the discriminator networks in the SkipGANomaly model. The generator consists of two components, an encoder and a decoder. An encoder network containing seven individual downsampling blocks, which takes an image with dimensions of 256x256x256 as input and compresses it to a 1x1x512 latent region. This region characterises the bottleneck of the whole generator network representing the learned encoded latent space representation. The decoder network then takes this encoded representation and aims to recreate the original data with the dimensions of a 256x256x3 matrix. As in the original GANomaly method, the encoder blocks utilize LeakyReLU activation while the Decoder network uses the ReLU activation function returning a generated image after a Tanh activation of the last layer. Kernel size and strides used in the convolutional layers are 4x4 and 2x2 respectively. After batch normalisation of each convolutional layer (using the same parameters as in the original GANomaly method), every encoder

[21] Named after Rudolf Lipschitz, Lipschitz constant is a minimum value where a function's first derivative is bounded by.
[22] Publicly available GitHub repository for GANomaly implementation https://github.com/samet-akcay/skip-ganomaly

block is skip-connected with the corresponding decoder block. It is where some of the layers in the neural network are skipped and the matching outputs of the encoder layers are fed to the decoder layers as inputs. The generator network ultimately returns a single output, the generated image $\hat{x}$.

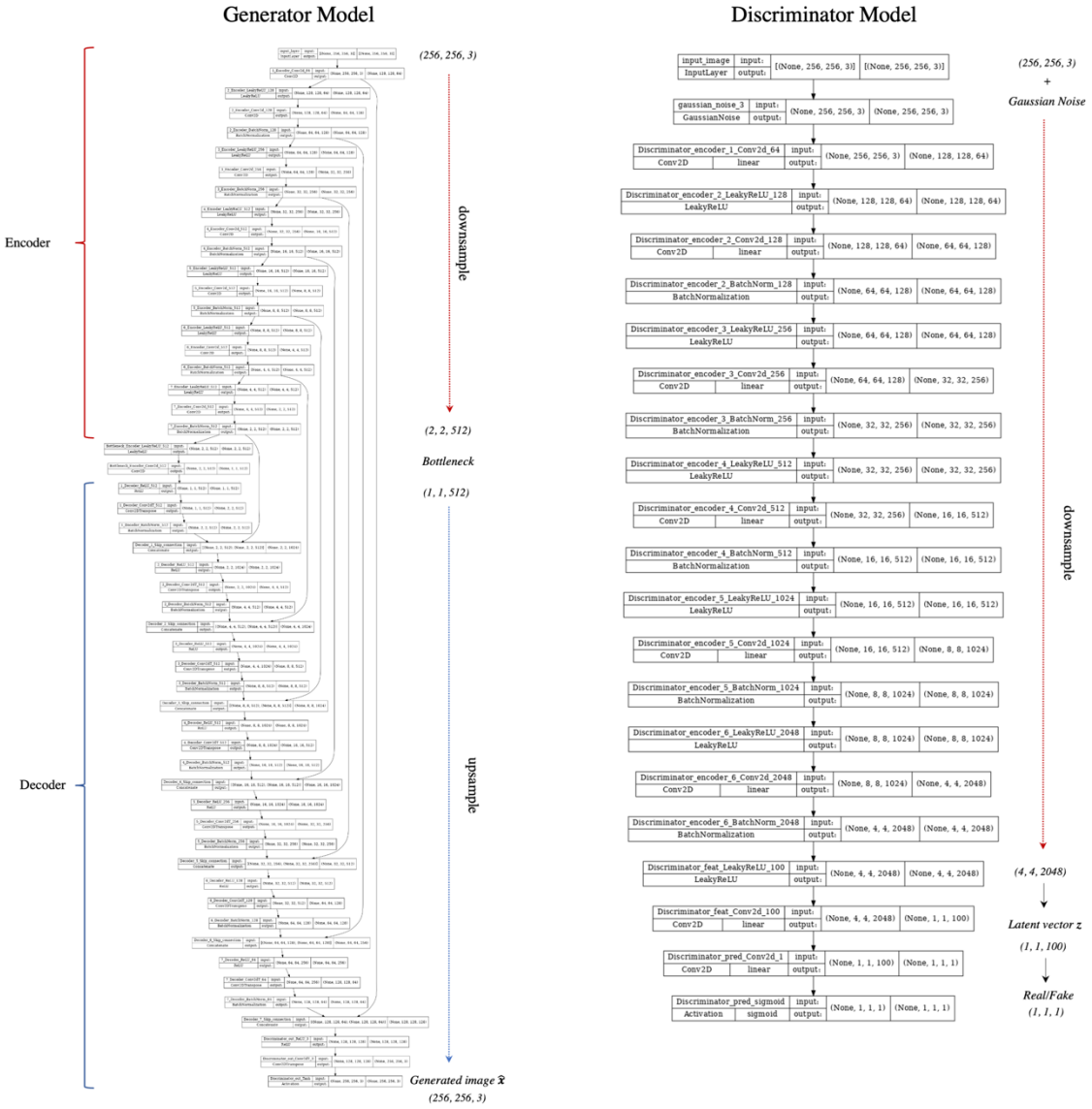


Figure 3.4. Original SkipGANomaly architecture for the generator and the discriminator networks respectively. The generator network comprises of one encoder network with seven downsampling blocks and one decoder network with seven upsampling blocks with skip connections between corresponding encoder-decoder pairs. The generator returns single output which is a reconstructed image $\hat{x}$. The discriminator network has seven downsampling blocks going all the to 4x4x2048 space and returns two outputs, latent vector (z or $\hat{z}$) and prediction probability in the form of real/fake classification.

Like in the generator's encoder network, the discriminator network is made up of seven downsampling blocks ending with a sigmoid activation function. The network takes an input image (real or generated) and compresses it to a 4x4x2048 matrix followed by a latent vector with dimensions of 1x1x100 (z, $\hat{z}$). In the end, the Discriminator network generates two outputs: the latent vectors and a real/fake classification after sigmoid layer.

Table 3.3 depicts the number of parameters in the generator and the discriminator network respectively.

Generator	Discriminator
Total params: 54,423,811 Trainable params: 54,413,955 Non-trainable params: 9,856	Total params: 47,992,896 Trainable params: 47,984,960 Non-trainable params: 7,936

TABLE 3.3. The number of total, trainable and non-trainable parameters for the generator and the discriminator networks respectively in the original SkipGANomaly method.

Similar to the GANomaly method, different sets of custom loss functions are used for the generator and discriminator models respectively as proposed in the original paper (Akçay et al., 2019). The loss function for the Generator is the sum of three different loss functions (Adversarial loss, Contextual loss, and Encoder loss) with corresponding weighting parameters ($\omega_{adv} = 1, \omega_{con} = 50, \omega_{enc} = 1$) that are responsible for adjusting the importance of the individual losses on the overall objective function. Adversarial loss used in the generator objective function is modified to become the original adversarial loss proposed by Goodfellow et al., (2014) (equation 3.7). Contextual and encoder losses are the same as in the original GANomaly method (equations 3.3 and 3.4), one difference being that the latent representations z and $\hat{z}$ extracted from x and $\hat{x}$ used in the encoder loss come from the intermediate convolutional layer of discriminator. Additional constraint is introduced in the discriminator objective function, by adding encoder loss to the overall adversarial loss (equations 3.8).

$$\mathcal{L}_{adv} = \mathbb{E}_{x \sim p_X}\left[log\left(1\text{-}D\big(G(x)\big)\right)\right] \tag{3.7}$$

$$\mathcal{L}_{D} = \mathbb{E}_{x \sim p_X}\big[log\big(D(x)\big)\big] + \mathbb{E}_{x \sim p_X}\left[log\left(1\text{-}D\big(G(x)\big)\right)\right] + \mathbb{E}_{x \sim p_X}\|G_E(x) - E(G(x))\|_2 \tag{3.8}$$

Both Generator and Discriminator models use Adam optimiser with learning rate α = 0.0002, and hyper-parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

3.6 Masked SkipGANomaly

After the initial results acquired from the original SkipGANomaly, it was obvious that the model was only copying the information of the input image and pasting it into the reconstructed image. Here we attempt to unravel the concept of whether the autoencoder network memorises the input samples and not target domain distribution, by corrupting the training image with a grid mask and trying impainting and interpolating the missing puzzle piece during training and inference. As shown in figure 3.5, we train the network exclusively on clean data and force the model to reconstruct the image patches that are masked with 128x128 blocks with a fault-free version of the missing image region. The network uses the original SkipGANomaly architecture and hyper-parameters and is trained for 200 epochs.

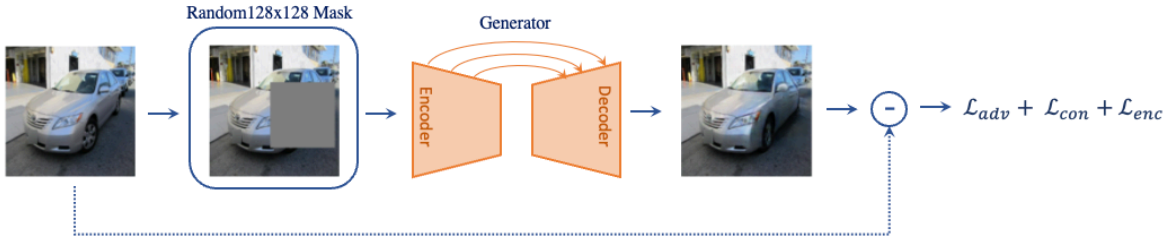


Figure 3.5. Train the generator with a skip-connected autoencoder network of undamaged cars corrupted with random 128x128 masks.

4 RESULTS

4.1 Introduction

This chapter covers the results of each method described in the previous chapter. In order to quantitatively compare our results between the methods we use a metric called The Fréchet Inception Distance score (FID). FID is a metric that measures the distance between the feature vectors for real and synthetic images (Heusel et al., 2018). FID score uses one of the best-performing image classification algorithms, the Inception v3 model (Szegedy et al., 2015), to categorise the synthetic images as one of the 1,000 known objects. FID scores combine confidence in both the quality and the diversity of the generated images. While the quality part assesses the class predictions of each image, the diversity evaluates the integral probability of those predictions. Unlike the reconstruction loss metric, FID scores don't capture the comparison between real and fake images. This score metric's primary objective is to evaluate the quality of generated images based on a statistical comparison between the collection of synthetic images and real images. These statistics are calculated by the last activation layer in the Inception v3 model that captures the specific features of the input image. Calculating the mean and covariance of the input images, the activations are summarised as a multivariate Gaussian. The difference between these two Gaussians (synthetic and real-world images) is then measured by the Fréchet distance also known as the Wasserstein-2 distance (Heusel et al., 2018). FID scores are computed by loading the mentioned pre-trained Inception v3 model and removing the output layer of the model. The feature vector of an image is represented as 2,480 activation features since the output layer has the same number of activations. The score is calculated using the equation below:

$$d^2 = \|\mu_1 - \mu_2\|_2^2 + Tr\left(C_1 + C_2 - 2\sqrt{C_1 * C_1}\right) \quad (4.1)$$

where,

- μ_1 and μ_2 are the means of the real and generated image feature vectors
- C_1 and C_2 are the covariance matrix for the real and generated feature vectors
- Tr refers to the trace linear algebra operation where the elements along the main diagonal of the square matrix are added

As a quantitative metric, images with lower FID scores indicate higher-quality images and the higher score values represent lower-quality images.

4.2 Method 1 Results – Original GANomaly method

As mentioned in section 3.2, the first experiment was conducted using the original GANomaly framework, with all the hyperparameters imitating the ones mentioned in the paper by Akcay et al., (2018). The paper results were acquired using the benchmark datasets i.e., MNIST[23] and CIFAR-10[24] datasets. As seen in figure 4.1, both datasets are very uniform and structured in a way that is easy to train and test. We train the same model using a much more complex real-world dataset containing car images, namely, the Nugen dataset (section 3.1). Hence, we anticipated some of the challenges represented in our results.

Figure 4.1. (left) Sample images from MNIST dataset of handwritten digits in 28x28 format, (right) sample images of CIFAR-10 dataset of colour images in 10 different classes in 32x32 format (https://www.cs.toronto.edu/~kriz/cifar.html).

The original GANomaly method was trained for 300 epochs with 385 steps each. The weights are initialized to be centred around zero with a standard deviation of 0.02. This stabilises both the generator and the discriminator by preventing the model gradients from vanishing or exploding. At the end of each epoch model performance was evaluated via, (1) mean generator and discriminator loss values for the epoch with their standard deviations, (2) the generator and discriminator loss plot for the current epoch, (3) all-time generator and discriminator loss plot up until the current epoch, and (3) generated images for training and validation dataset. Even if the loss plots show convergence, it is always a good idea to check the reconstructed images to detect any possible mode collapse. Mode collapse is where the generator manages to find few samples that the discriminator cannot distinguish as either fake or real, resulting in the generator giving out only those few samples and not creating any new instances.

---

[23] MNIST database is a large database of black and white handwritten digits (60,000 training images and 10,000 testing images) in 28x28 format with uniform background widely used in training and testing in machine learning (https://en.wikipedia.org/wiki/MNIST_database).

[24] CIFAR-10 database is a collection of images of around 60,000 samples of 32x32 colour images in 10 different classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks) (https://en.wikipedia.org/wiki/CIFAR-10).

Figure 4.2 demonstrates the all-time generator and discriminator loss plot. Discriminator loss consists of two parts, the probability of the image being real and the probability of the image being fake. The total discriminator loss is measured by adding these individual losses together and dividing them by 2. Ideally, the discriminator loss should be around 0.5 for one instance, which means that the discriminator is now guessing whether the image is real or fake. The generator loss consists of three parts, adversarial loss, contextual loss and encoder loss. Ultimately, the generator loss should decrease over time, in an ideal case converging with the discriminator loss.

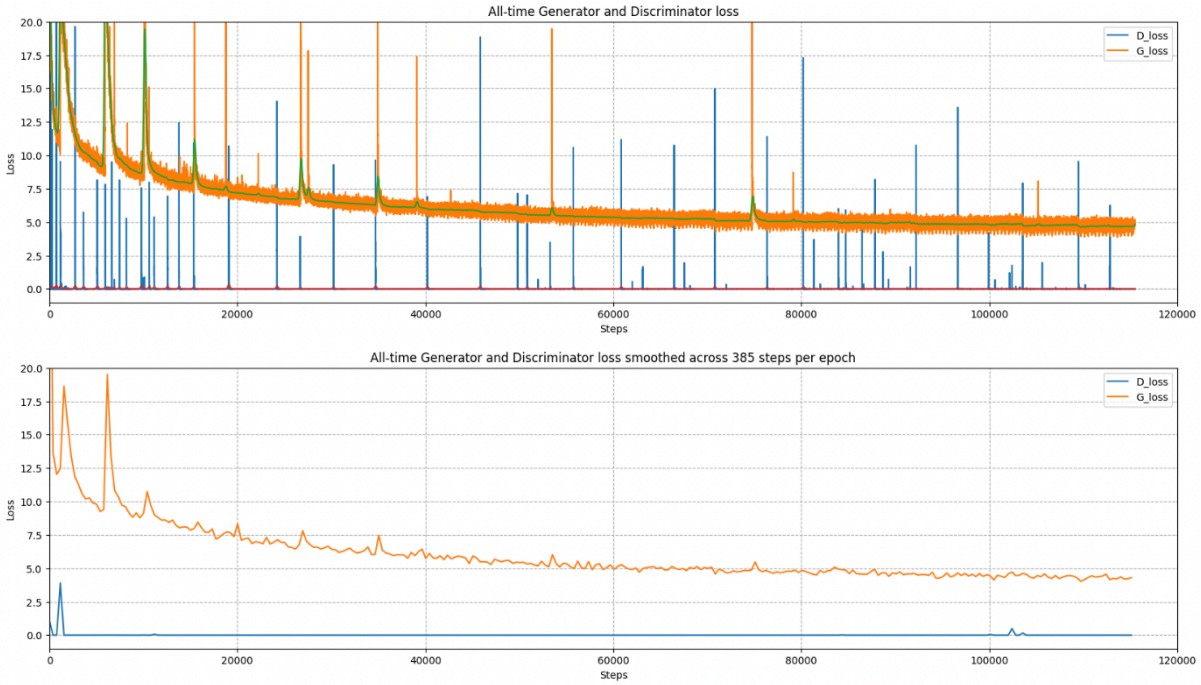


Figure 4.2. All-time generator and discriminator loss plot (G_loss, D_loss) for original GANomaly method. (top) per step, (bottom) per epoch, smoothed across 385 steps. The model was trained for 300 epochs, with 385 steps each (115500 total steps).

As seen in figure 4.4 the average discriminator loss is around zero, which means the discriminator is doing a better job of distinguishing between real and fake images. This seems to be a known problem of GANs, where the generator is doing a bad job of creating samples that match the real data distribution. To sum it up, it is important to define a loss function for the discriminator in a way that the whole GAN system has losses balanced out. When the discriminator approaches zero it is an indication of a failure mode where no more learning takes place. It means that the discriminator becomes so strong that its adversary, the generator, cannot improve itself.

Despite little help from the discriminator through backpropagation, generator loss still decreases with each iteration, thanks to the contextual loss function embedded in the overall generator objective function. As seen in figure 4.3 the encoder and adversarial losses provide little to no gradient signal for the generator. As observed from the generated images (figure 4.5), however, this is insufficient to produce realistic images, as the images are too blurry. There are two reasons for this behaviour. The first reason being the contextual loss that is calculated using MSE loss. Since the other losses provide

little to no gradient to the network, the generator tries to perform an image-to-image mapping with normal data samples only under the minimisation of the contextual loss. MSE loss is known to have the disadvantage of reconstructing blurry images, resulting in higher residuals for clean structures. The other reason for blurry reconstruction is the latent dimension (bottleneck), which is way too small. In essence, the autoencoder model is trying to compress a huge number of values into a small two-dimensional latent space. Autoencoders generally are known for generating blurry outputs due to such small latent dimensions making it hard for the information from the input to bypass the small bottleneck. By increasing the bandwidth of the bottleneck (latent dimension) we might prevent further data leakage and get higher reconstruction quality. For this reason, as discussed in section 3.3, the latent dimension is increased to 512 in our next experiment.

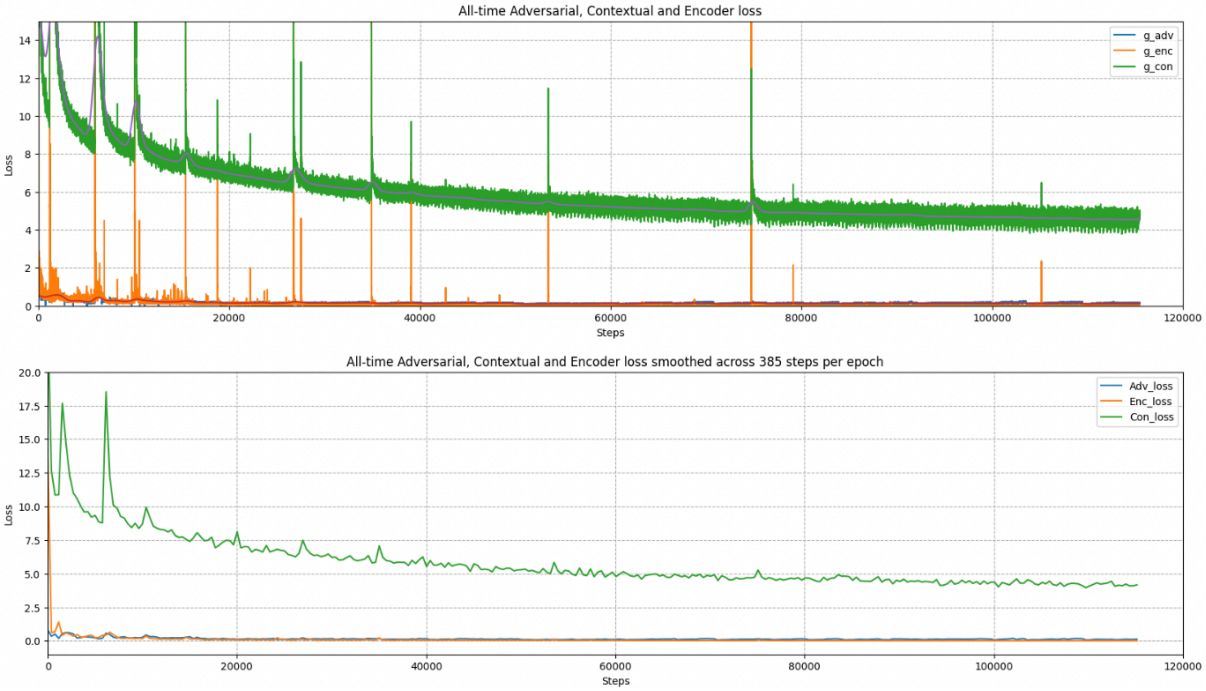


Figure 4.3. All-time generator losses plot (*g_adv – Adversarial loss, g_enc – Encoder loss, g_con – Contextual loss*) for original GANomaly method. (top) per step, (bottom) per epoch, smoothed across 385 steps.

Figure 4.5 depicts samples of generated images for the training (undamaged) and test sets (damaged) respectively. As seen from the figure below, the reconstructed images from the training set tend to fair much better than the images in the test set, which is further confirmed by the FID scores depicted in figure 4.6. Looking at the reconstructed test images, we can clearly notice that the model is trying its best to replace the defective structures with clean ones. Some of the images in figure 4.5 show where the foreign objects such as measuring tape, human hand as well as scribbles on the car surface are removed successfully. In one instance the model is attempting to reconstruct the missing back bumper while in another the dent on the right bumper is being completely removed. However, it doesn't take from the fact that the generated images are of poor quality with high blur and some of the structures on the car being lost during the reconstruction.

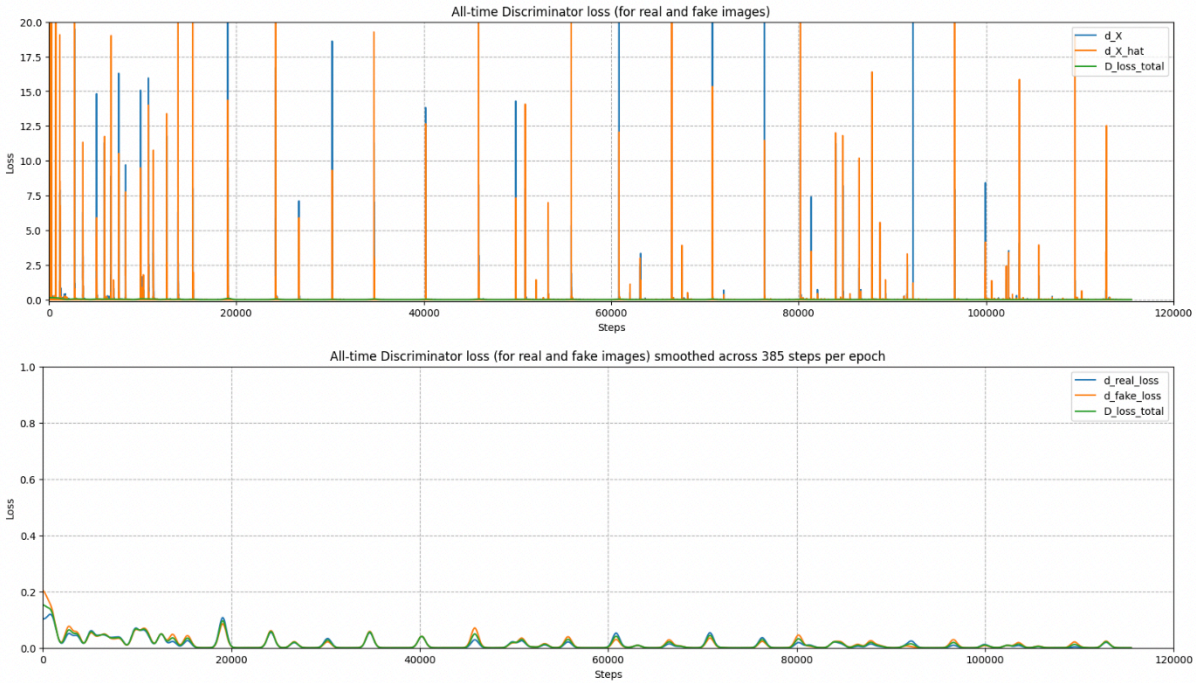


Figure 4.4. All-time discriminator loss plot (*d_x – real, d_x_hat – fake*) for original GANomaly method. (top) per step, (bottom) per epoch, smoothed across 385 steps.

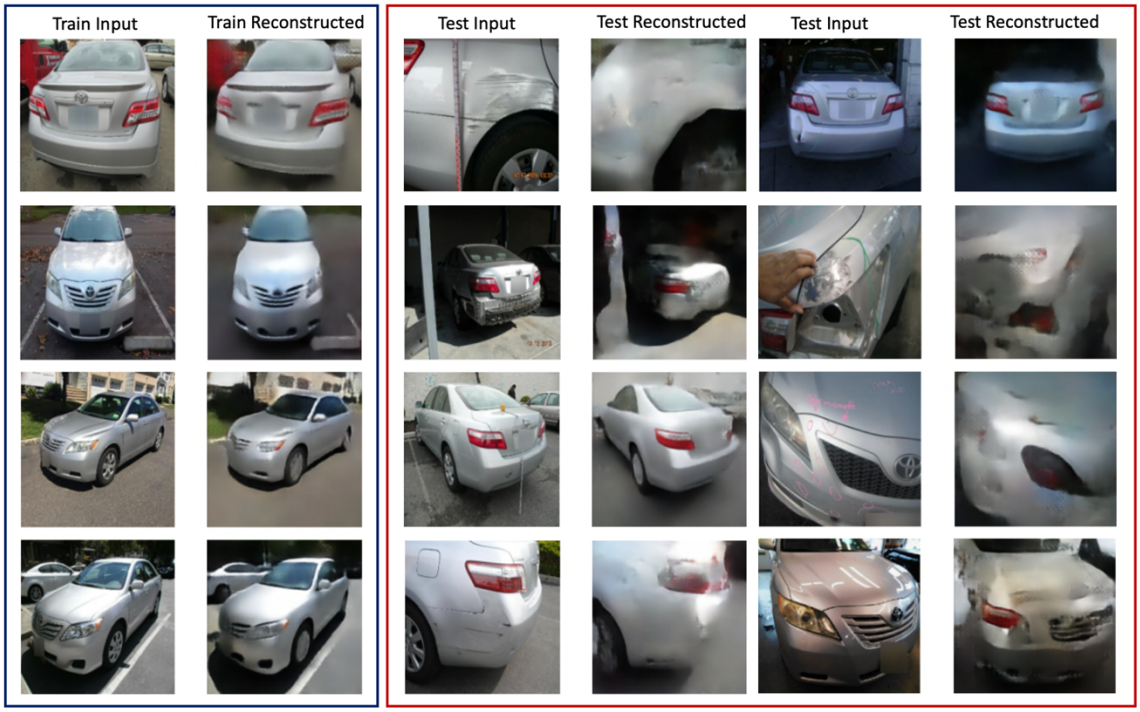


Figure 4.5. Generated images for training (blue) and test sets (red) from the original GANomaly method.

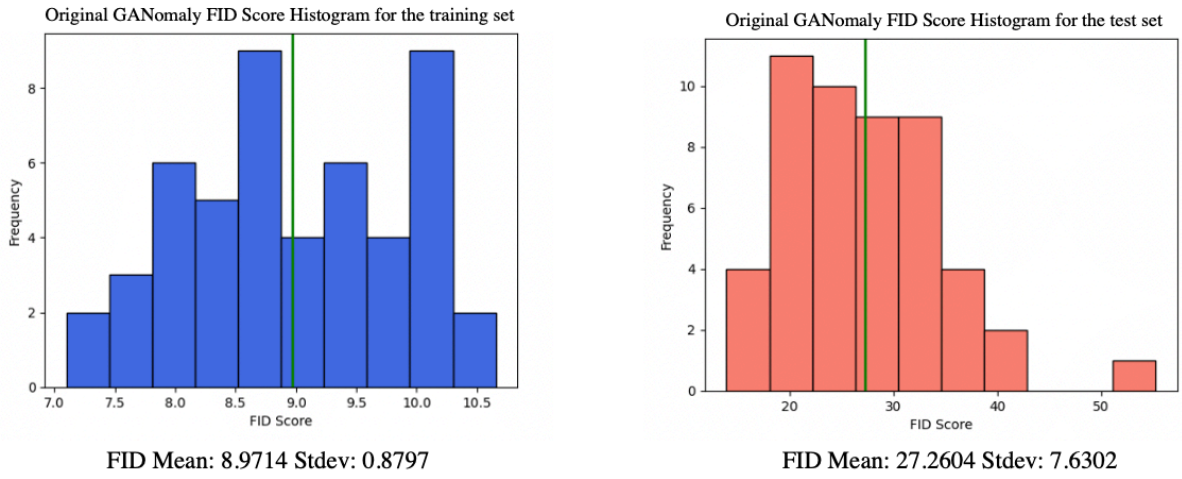


Figure 4.6. FID Score histogram for 50 randomly sampled reconstructed images from training and test set respectively (original GANomaly method).

4.3 Method 2 Results – Modified GANomaly

The original GANomaly approach in section 4.1, provides us with the baseline framework upon which we might expand. Due to the characteristics of our dataset, the model contains a number of flaws, which we want to mitigate using a variety of methods in this next experiment. As discussed in section 3.3 we address some of the weaknesses of the original GANomaly method with several techniques uncovered through an extensive literature review (larger kernel size, larger latent dimension, soft and noisy labels, adding noise to the discriminator, TTUR, and modifying adversarial loss function). Once again, we train our model for 300 epochs with 385 steps each.

Figures 4.7, 4.8 and 4.9 depict the all-time losses for the GANomaly method with the modified features. As explained in the previous section, the discriminator in the original GANomaly method was very effective at distinguishing between real and fake images. To make the discriminator's task more difficult we add soft and noisy labels to the discriminator loss function as well as an additional Gaussian noise at the top of our discriminator's convolutional stack. As a result of these improvements, we are able to avoid the discriminator from approaching zero loss very rapidly which can prematurely kill all learning processes early on. Figure 4.9 clearly shows that the discriminator loss for neither real nor fake samples never reach the zero value.

The generator gets most of its gradient signal from contextual loss and modified adversarial loss (figure 4.8). As discussed in section 3.3 the adversarial loss was slightly modified to provide an additional penalty for the generator's overall objective (equation 3.6).

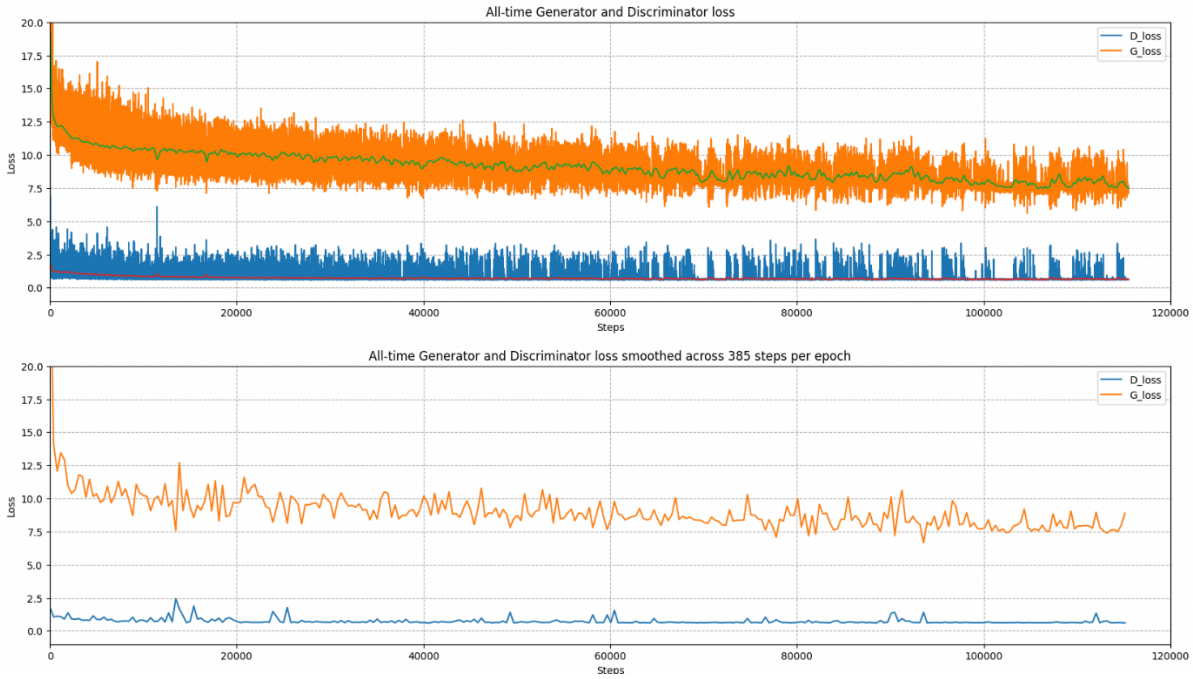


Figure 4.7. All-time generator and discriminator loss plot (G_loss, D_loss) for modified GANomaly method. (top) per step, (bottom) per epoch, smoothed across 385 steps. The model was trained for 300 epochs, with 385 steps each (115500 total steps).

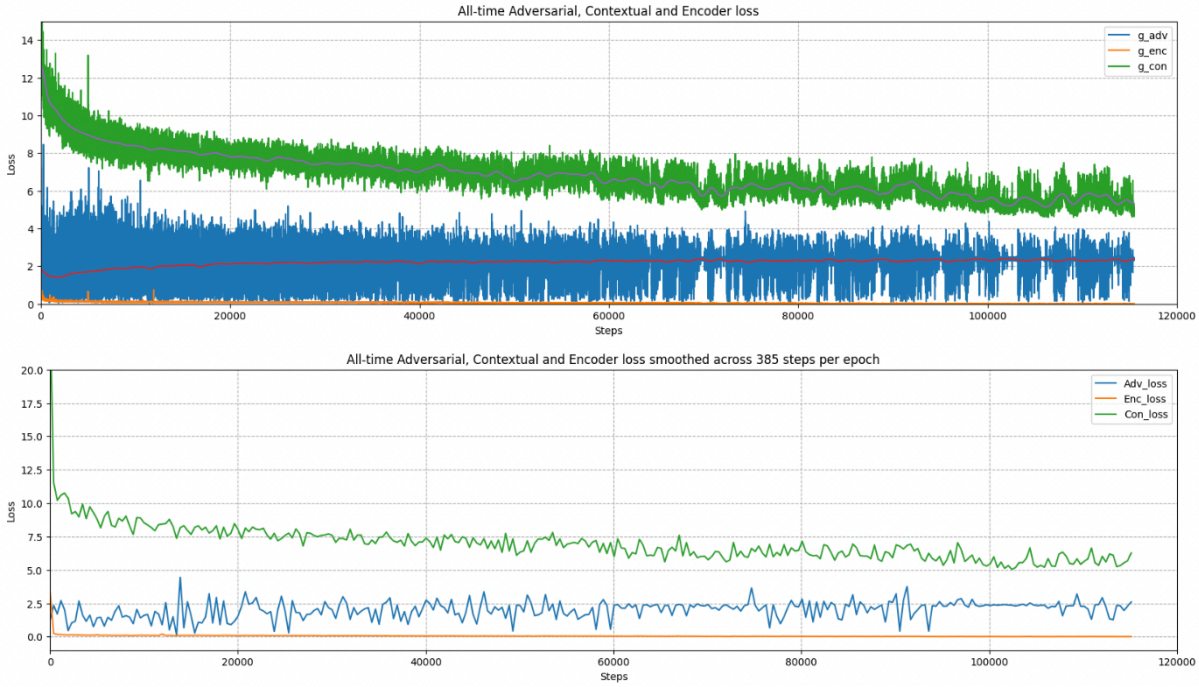


Figure 4.8. All-time generator losses plot (*g_adv – Adversarial loss, g_enc – Encoder loss, g_con – Contextual loss*) for modified GANomaly method. (top) per step, (bottom) per epoch, smoothed across 385 steps.

Figure 4.9. All-time discriminator loss plot (*d_x – real, d_x_hat – fake*) for modified GANomaly method. (top) per step, (bottom) per epoch, smoothed across 385 steps.

As seen in figure 4.10, the generated images for the test set are of much better quality in comparison to the results obtained from the first method, but they still represent a high level of blur. From the figure, we can see the model making a better effort in replacing the defective structures with the non-anomalous ones, where in one instance the network tries to put the falling bumper back together, or in another where the right-side door damage is almost completely removed. Additionally, the foreign objects obscuring the car image such as measuring tape are somewhat removed. However, in another instance, the model fails to reconstruct the missing left side-view mirror. Despite of all its successes and failures, it is clearly visible that the reconstructed test images from the modified GANomaly method are of much better quality than the ones generated from the original method (figure 4.5) which is further confirmed by the FID scores depicted in figure 4.11. When comparing with figure 4.5, we can see that the generated images retain most of the structures of the car during the reconstruction. The FID scores are slightly higher for the training set, however for the test set, they are much lower than the ones acquired from the original GANomaly method. This proves that the modifications made to the original model have worked and we are moving in the right direction.

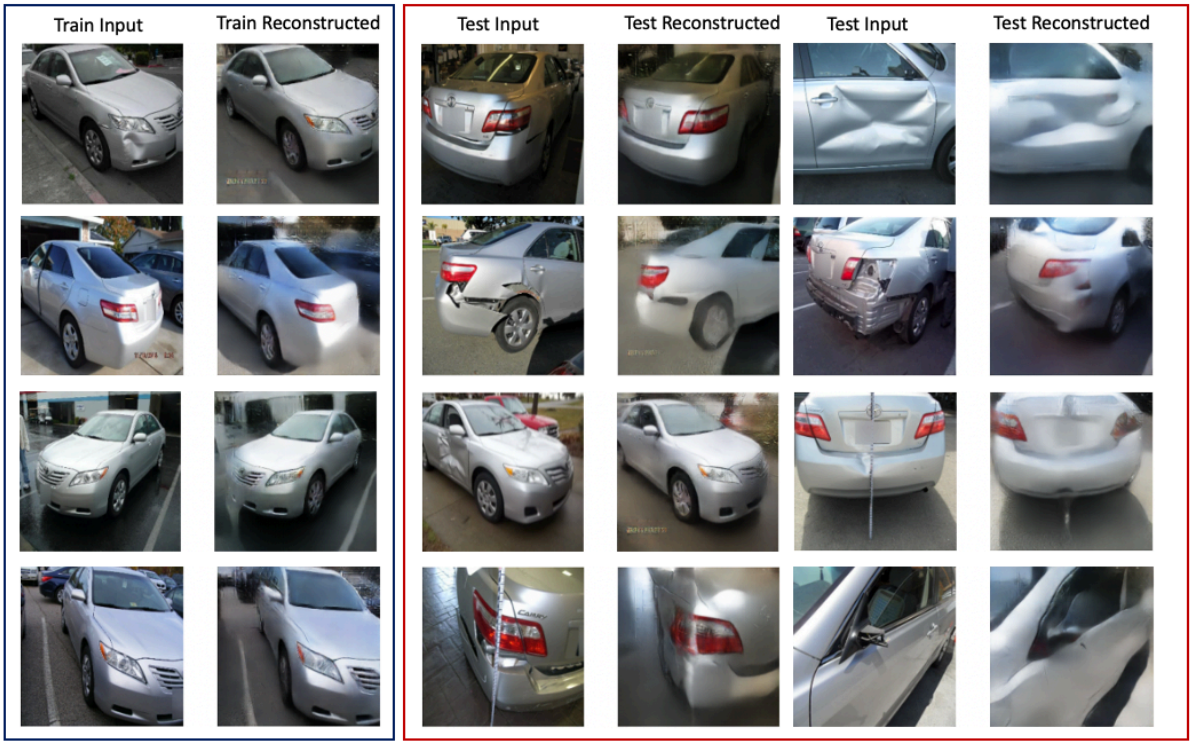
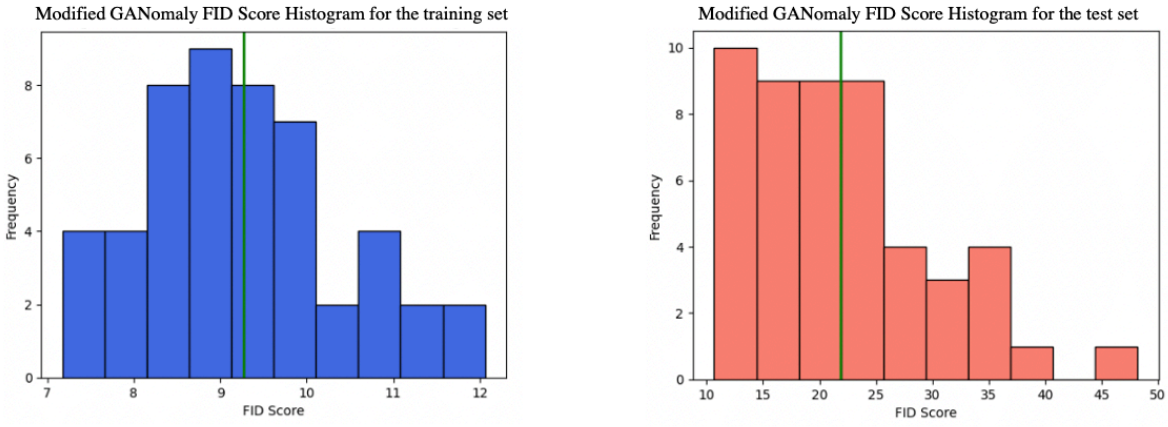


Figure 4.10. Generated images for training (blue) and test sets (red) from the modified GANomaly method.



FID Mean: 9.2694 Stdev: 1.1215

FID Mean: 21.8322 Stdev: 8.0764

Figure 4.11. FID Score histogram for 50 randomly sampled reconstructed images from training and test set respectively (modified GANomaly method).

4.4 Method 3 Results – GANomaly with Spectral Normalisation

As discussed in section 3.4, the third experiment was conducted by applying spectral normalisation on the discriminator's convolutional kernels. Everything else is kept the same, i.e., model architecture (figure 3.2), trainable parameters (table 3.2) and hyper-parameters (section 3.3).

Figures 4.12, 4.13 and 4.14 depict the all-time losses for the GANomaly method with the spectral normalisation.

Figure 4.12. All-time generator and discriminator loss plot (G_loss, D_loss) for modified GANomaly method spectrally normalised discriminator model. (top) per step, (bottom) per epoch, smoothed across 385 steps. The model was trained for 300 epochs, with 385 steps each (115500 total steps).

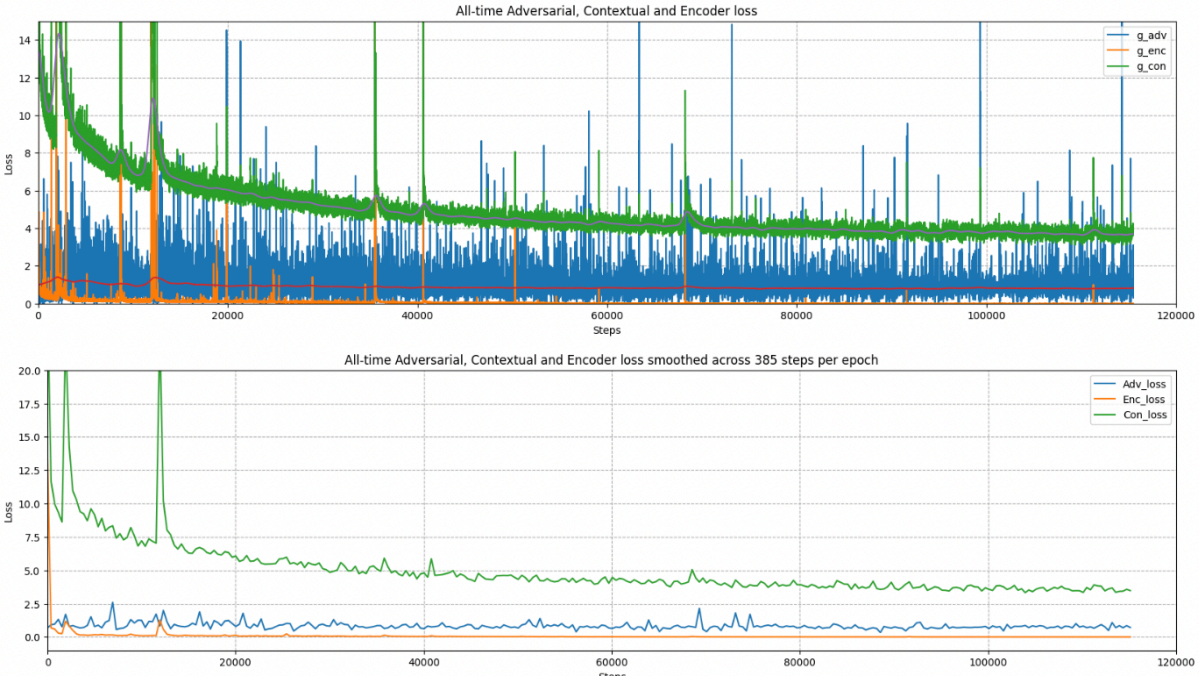


Figure 4.13. All-time generator losses plot (*g_adv – Adversarial loss, g_enc – Encoder loss, g_con – Contextual loss*) for modified GANomaly method with spectrally normalised discriminator model. (top) per step, (bottom) per epoch, smoothed across 385 steps.

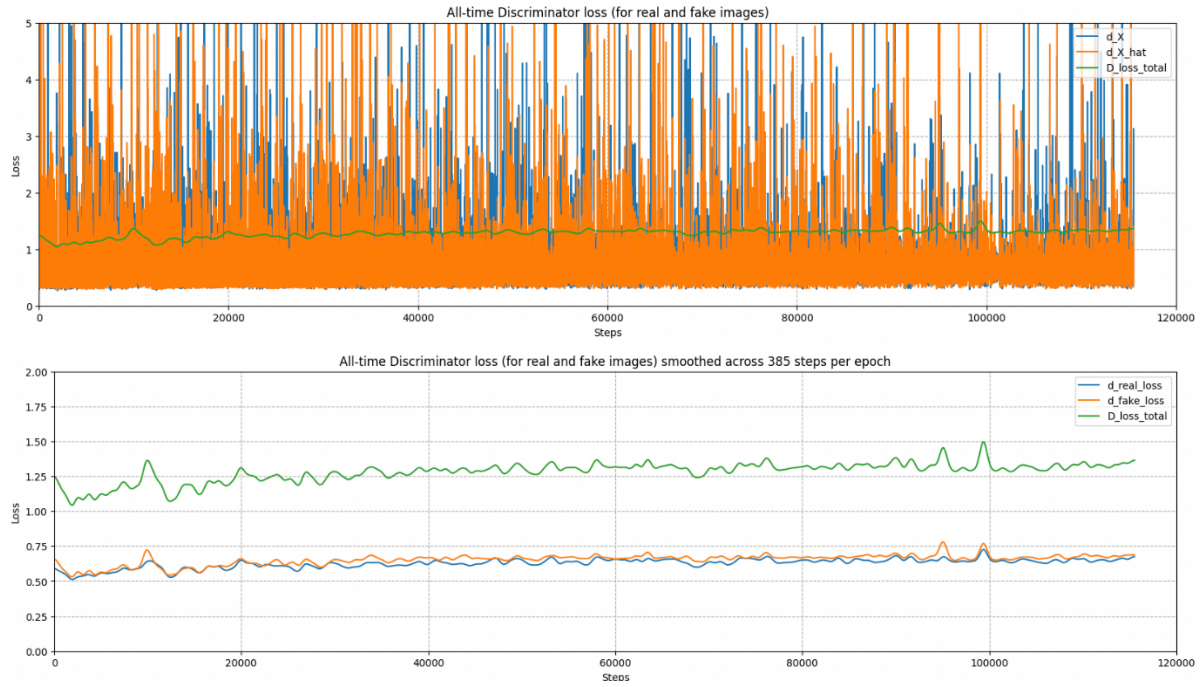


Figure 4.14. All-time discriminator loss plot (*d_x – real, d_x_hat – fake*) for modified GANomaly method with spectrally normalised discriminator model. (top) per step, (bottom) per epoch, smoothed across 385 steps.

As seen from the loss plots, the generator loss converges with the discriminator loss much more rapidly. Additionally, unlike the previous method, encoder loss provides a much higher gradient signal to the generator at the beginning of the training. Apart from discriminator losses for real and fake images never reaching zero, the total loss increases slightly as training commences.

Qualitatively, when examining the generated images (figure 4.15) we can clearly notice the quality of the training images being much higher that its previous counterparts. However, in both the training and test set, apart from the blurry reconstruction a noise is present in the generated images. Some of the reconstructed images for test set show good results when it comes to removing damage and/or artifacts such as foreign objects. Nevertheless, the quality is not of the level we would like, as the network still struggles to infer the finer details of the car images.

Figure 4.16 presents the FID score performances obtained with the newer approach with spectral normalisation and compares the metric between training and test sets. The mean training FID score for both training and test is substantially lower than in any previous method (Mean FID train: 3.9613, test: 14.9134), which further highlights the benefits of using spectral normalisation in the discriminator model.

Figure 4.15. Generated images for training (blue) and test sets (red) from the modified GANomaly method with spectrally normalised discriminator model.

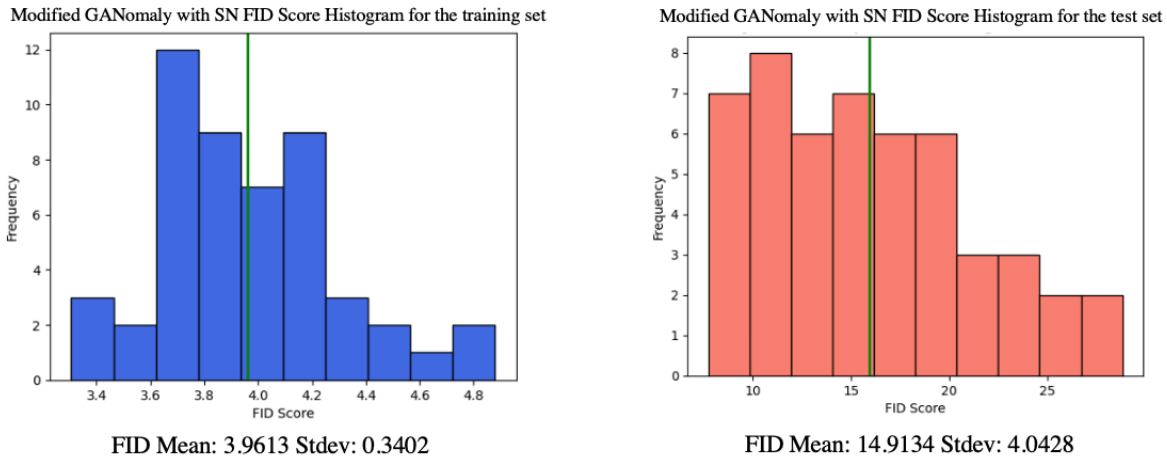


Figure 4.16. FID Score histogram for 50 randomly sampled reconstructed images from training and test set respectively (modified GANomaly method with spectrally normalised discriminator model).

4.5 Method 4 Results – Original SkipGANomaly

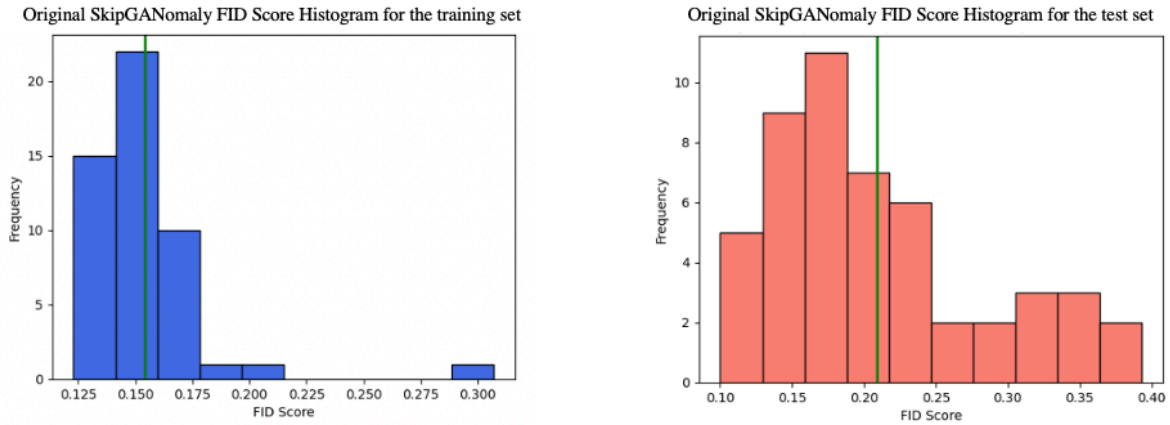
The GANomaly methods discussed in previous sections all suffer from blurry low-quality reconstructed images. As mentioned before the autoencoder component of the generator network in the reconstruction-based GANomaly method is trained exclusively on the undamaged car images in order to regularise the reconstruction towards the normal class. Due to the bottleneck in the form of a small latent space, the network is constrained to learn a compressed representation of the training data. Previous literature on hourglass CNN architectures without skip connections has been criticised for

their tendency to produce blurry images (Zhao et al., 2018) (Bergmann et al., 2019). Hence, the authors of the original GANomaly framework proposed an improved version of the method with a skip-connected autoencoder network in order to enhance the sharpness of the reconstruction (figure 3.3).

However, the reconstructed images reveal that the SkipGANomaly framework suffers from a serious flaw. Since there is no information loss in the network, the model provides a straightforward copy-and-paste procedure that yields an accurate reconstruction of an arbitrary image (figure 4.18). As a result, whatever goes into the generator comes out on the other side as reflected by the FID scores (figure 4.19). As seen in figure 4.17 the model converges within 10 epochs. This in turn made us question the hypothesis behind the framework, which is the compression induced by the bottleneck allows the reconstruction that lies on the non-anomalous manifold. It is true that skip connections allow the preservation of high-frequency information by bypassing the bottleneck. Ideally, the GANomaly network is supposed to preserve non-anomalous structures while modifying those that are not. But the SkipGANomaly model fails to differentiate between those two concepts (normal and abnormal samples). Therefore, in the next method, we propose introducing a synthetic corruption in the form of a square mask in order to formulate the reconstruction task as an image completion problem.

Figure 4.17. All-time generator and discriminator loss plot (G_loss, D_loss) for original SkipGANomaly method. (top) per step, (bottom) per epoch, smoothed across 385 steps. The model was trained for 300 epochs, with 385 steps each (115500 total steps).

Figure 4.18. Generated images for training (blue) and test sets (red) from the original SkipGANomaly method.



FID Mean: 0.1543 Stdev: 0.0274

FID Mean: 0.2088 Stdev: 0.0741

Figure 4.19. FID Score histogram for 50 randomly sampled reconstructed images from training and test set respectively (original SkipGANomaly method).

4.6 Method 5 Results – Masked SkipGANomaly

As seen in section 4.5, training the original SkipGANomaly network on normal samples resulted in a full reconstruction of the arbitrary input image. The original SkipGANomaly network operates under the assumption that the compression produced by the bottleneck is sufficient for regularised reconstruction to lie on non-anomalous image space. Nevertheless, the generator network with the

autoencoder reconstructs the defective structures as it is not explicitly constrained to not reproduce abnormal content. In order to mitigate this issue, as an extension of the SkipGANomaly method, we propose to iteratively project arbitrary input towards the clean distribution of images through the introduction of synthetic anomalies, such as random 128x128 mask. That way we formulate the original reconstruction task of the SkipGANomaly method as an image completion problem. Haselmann et al., (2018) proposed a similar work where they were able to train a deep convolutional neural network to complete images whose centre regions are cut out. As a result, by training the network exclusively on clean data, they were able to complete the image patches that are masked with a fault-free version of the missing image region.

Figure 4.20. All time generator and discriminator losses for masked SkipGANomaly method smoothed across 385 steps per epoch.

Looking at the all-time loss plots of generator and discriminator losses, we can see that training for 4000 steps which is equivalent to 100 epoch is sufficient for the model to find its equilibrium under the current hyper-parameters (figure 4.20). It is when the discriminator loss for both real and fake losses both reach the value of 0.5, which is an indication that discriminator cannot distinguish between input and generated image. Training it for the next 100 epoch just overfits the model as seen in the slight increase in the generator loss which we would like to avoid.

Figure 4.21 shows some of the results of reconstructed images from the masked SkipGANomaly method. As seen in the images, the model successfully completes the masked patches with the fault-free version of the car region. The model is able to reconstruct the missing structures of the damaged cars with corresponding non-anomalous structures such as tail lamps, back bumpers etc. The network generates higher-resolution images with an understanding of the individual components in the car structure that goes in the masked regions.

Figure 4.21. Generated images for training (blue) and test sets (red) from the masked SkipGANomaly method.

5 DISCUSSION

5.1 Evaluation of Objectives

This project was able to achieve unprecedented results through the use of state-of-art GAN frameworks for anomaly detection trained on a complex and messy Nugen dataset. Initial Exploratory Data Analysis (EDA) of the dataset revealed similar challenges seen on real-world datasets if not more, however, further data processing might yield much better results in the future (i.e., background removal).

The first three methods proposed by this research are the GANomaly framework originally proposed by Akcay et al., (2018) and their variations. Upon detailed inspection of the results, we concluded that neither of the models yielded desirable quality results to meet the requirements of the objectives in this study. The fourth method, the original SkipGANomaly (Akçay et al., 2019), on the other hand, suffered from a serious flaw, where the model penalised anomaly detection due to it being trained to perform identity mapping on normal samples. Therefore, the final method proposed by this research was a variation of the original SkipGANomaly method, where we reformulated the reconstruction task as an image completion problem. That way enabling the model to reconstruct the missing structures of the damaged cars with corresponding anomaly-free structures. Despite its success in producing much better results than GANomaly, the method relied on the assumption that the damages are entirely contained within the masked region, which limits the practical usage of the model. Therefore, further study and expansion of this model are recommended for future work. Due to time limitations, we were unable to explore the sixth experimental part, where we propose using the grid-mask method. Figure 5.1 demonstrates the proposed method, where we create an alternating grid mask for the arbitrary image, upon which one set of the reconstructed image is generated. By concatenating the generated masked regions from both images, we can use the adversarial, contextual and encoder losses to backpropagate through the generator network.

Overall, we believe this research provides substantial proof of concept demonstrating the potential GANs have to offer in unsupervised anomaly detection that can be further researched by Tractable AI given the recommendations provided in the future work section of chapter 6. Our final method provides enough proof that the adversarial training can in fact discern the distribution and particular features of the uncorrupted clean samples to be later inferred on examples with anomalies. We were able to show that by introducing synthetic anomalies to the arbitrary inputs, in the form of masked patches, we can significantly improve the anomaly detection accuracy for the skipped autoencoder architecture, eventually outperforming the conventional GANomaly method.

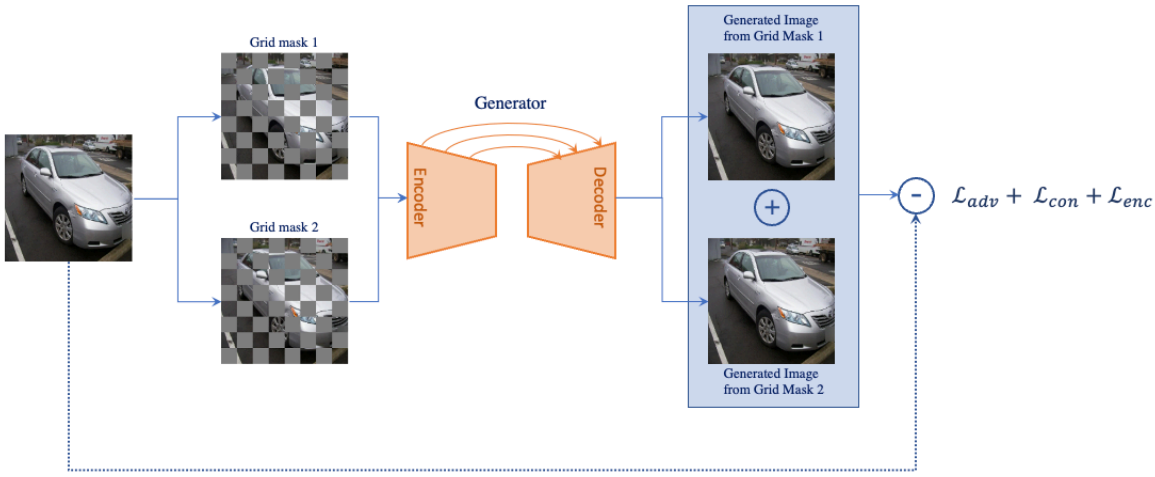


Figure 5.1. Proposed grid mask SkipGANomaly method. The skip-connected autoencoder component of the generator receives two images which are alternating grid-mask images of the arbitrary input and reconstructs one set of images. Upon concatenating the generated masked regions, we can use the three loss functions proposed in the original framework to backpropagate through the generator network.

5.2 Answering the Research Question

The aim of this project was to help understand whether GANs can help detect damages in car images via unsupervised learning, as in without the need for labelled datasets. Although there exists subjectivity in the datasets used in GAN models, this work aims to remove the reliance on labelled datasets in an effort of reducing the labour and expense associated with acquiring such. There are definite results that could be taken from this study, and with deeper networks, hyper-parameter optimization, architectural modifications and novel approaches for objective functions, we believe the outcomes of this study can grow further. The acquired results clearly exhibited that damages in cars can definitely be detected via unsupervised adversarial learning on clean samples.

6 EVALUATION, REFLECTIONS, AND CONCLUSIONS

6.1 Acquired knowledge

Through this project, despite having no prior knowledge or experience in this emerging field of computer vision, I have made a great leap forward in knowledge. From learning the fundamentals of state-of-art GAN networks to understanding the benefits and drawbacks of training such models, I have acquired a greater understanding of this new cutting-edge domain. Through acquiring a deeper knowledge of hourglass CNN networks, I was able to understand the principles of DCGAN architecture and how it led to many advancements in GAN frameworks. By studying some of the most recent literature on anomaly detection via adversarial training, I have familiarised myself with regularised reconstruction-based approaches where an arbitrary input is iteratively projected towards the clean distribution.

Additionally, I have had an opportunity to learn how to pre-process data, build efficient data pipelines and train deep neural network models on professional servers using GPUs. I have also become more proficient in coding complex model architectures in TensorFlow API and Keras libraries, both of which played a pivotal role in this research, as they allowed me to allocate more resources to experimenting with different models. Overall, I acquired extensive knowledge in the GAN frameworks which feeds further to my passion for the field of computer vision.

6.2 Choice of objectives

Overall, the choice of objectives for the project proved to be ambitious, but we were able to yield some encouraging results within a reasonable amount of time. Given further study and experimentation we believe this study can be taken further and become one of the most promising unsupervised models towards car damage detection on real world datasets ready for deployment. To the best of our knowledge, prior works with similar frameworks were only performed on clean and uniform benchmark datasets. We were able to demonstrate the relevance of our approach with consistent assessment of reconstruction-based image completion method, by comparing its performance over a complex real-world Nugen dataset.

6.3 Project limitations and weaknesses

In this work, we attempt to detect car damages based on the reconstruction of a non-anomalous image from any arbitrary image through GAN training. The last method builds on a skip-connected convolutional autoencoder network for the generator model that relies on the reconstruction error between arbitrary input and its reconstructed version, estimated with the random masked

SkipGANomaly technique, to detect anomalies. We demonstrated the benefits of considering an autoencoder architecture equipped with skip connections, given that the training images are corrupted with random masked noise to avoid convergence towards an identity operator. This new approach performs significantly better in identifying all diverging samples that do not fit the regular clean data distribution, compared with traditional GANomaly and SkipGANomaly methods proposed by Akcay et al., (2018, 2019).

However, the method relies on the assumption that the damages are entirely contained within the masked region, which limits the practical usage of the model. As such, the framework reaches the limit of its underlying assumption, by not replacing all anomalous structures with clean content with sufficient high reconstruction error.

This study provides strong proof of concept for anomaly detection via adversarial training; however, providing a model ready for deployment is beyond the scope and time constraints of this project. In the next section, we provide some potential future works that might be performed to refine the framework, should sufficient resources be made available for such research.

6.4 Future Work

Going Deeper

The use of Residual Blocks as part of the encoder and decoder network within the generator module is a further alternative to the standard convolutional layers for the feature extractor. Over the years, deep neural networks have become the driving force behind the most powerful machine learning algorithms. Generally, neural networks with combined layers perform a given task, but as we stack more layers together while going deeper by increasing the depth of the network, the model performance can be substantially increased. He et al., (2015) introduced a novel deep neural network architecture called Residual Networks, which are 152 layers deep and contain skip connections that help with optimisation. Residual blocks are able to maintain the weights and continuously optimise and improve accuracy when some of the stacked layers in the neural network add zero value. Possible insights into our GAN models' behaviour, such as whether they are memorising training samples or not and how close they are to learning the target domain distribution, could be unravelled through feature extraction using Residual Blocks.

Wasserstein loss

Even though GANs are powerful generative models, they suffer significantly from training stability. Improving GAN training is a well-established fact that can be attained through the application of cutting-edge optimization methods like Wasserstein Generative Adversarial Networks (WGANs). (Arjovsky et al., 2017) . The original WGAN method enforces Lipschitz constraint on the critic via a

weight clipping, which can more often than not lead to undesirable results. To mitigate that problem Gulrajani et al., (2017) proposed a novel approach for clipping weights where the norm of the gradient of the critic is penalised with respect to its input. Apart from improving the stability of the training, WGAN provides a loss function that correlates with the quality of the reconstructed images. Wasserstein loss shifts away from the traditional concept of a discriminator which predicts the probability of the generated image being real/fake, to a critic model that evaluates the authenticity of an image. In essence, if the model's predicted probability distribution for an image deviates too greatly from the target probability distribution, the loss function will penalise the model. Through backpropagation, Wasserstein loss feeds the error from the previous batch into the discriminator and the generator to improve their performance on the next.

Background Removal

The original GANomaly (Akcay et al., 2018) and SkipGANomaly (Akçay et al., 2019) employ the reconstruction loss to calculate an anomaly score for a given test image $\dot{x}$ (equation 6.1).

$$\mathcal{A}(\dot{x}) = \lambda R(\dot{x}) + (1 - \lambda)L(\dot{x}) \tag{6.1}$$

where;

- $R(\dot{x})$ is the reconstruction score measuring the contextual similarity between the input and the reconstructed image bases on equation 3.2.
- $L(\dot{x})$ is the latent score measuring the difference between the latent vectors of input and the reconstructed image based on equation 3.3.
- λ is the weighing parameter controlling the relative importance of reconstruction and latent scores.

Due to the fact that original methods were trained on benchmark datasets with uniform backgrounds, the application of the anomaly score to our dataset was not feasible. The Nugen dataset contains images with varying backgrounds. Therefore, one direction for future work would be using the car images with the backgrounds removed using a publicly available pre-trained U-2-Net[25] model (Qin et al., 2020). The network borrows its general architecture from its predecessor U-Net, with a two-level nested U structure (Ronneberger et al., 2015).

Curated Dataset

Having a more curated dataset with unequivocally undamaged car images of good quality might further assist in improving the performance of the model. Even though the Nugen dataset did provide enough

[25] Open source GitHub repository for pretrained U-2-Net model (https://github.com/xuebinqin/U-2-Net)

traction for understanding the potential of GANs in car damage detection, it did possess additional challenges for the model in the form of artefacts and background noise.

Uncertainty-based anomaly detection (Monte Carlo Dropout)

The current model proposed by this research relies on the assumption that an anomaly map can be generated by subtracting an arbitrary input image from its reconstruction (aka residual-based anomaly detection). Collin and De Vleeschouwer (2020) introduce a similar skip-connected autoencoder network with a novel uncertainty-based anomaly detection. The technique proposes to use Monte Carlo dropout (MCDropout) during inference time, in order to generate an anomaly heatmap estimated by the variance between *N* reconstructions. The technique relies on the hypothesis that the higher reconstruction uncertainty will correlate well with anomalous regions on the image.

MCDropout is like regular dropout but applied during inference in order to get an estimate for model uncertainty on multiple test runs (Gal and Ghahramani, 2016). Dropouts are a regularisation technique used extensively in deep learning models to mitigate over-fitting (Srivastava et al., 2014). In layman's terms, at each training step, a different set of neurons are switched off (discarded), effectively creating a network with a different number of nodes and connections. Similarly, MCDropout makes the testing process noisy, by introducing a dropout rate that controls the probability at which a random fraction of the layer's outputs is retained. With different neurons switched off, a different set of networks are created. From the space of all available models, each model produces a different set of predictions. By averaging those predictions, we acquire the model's uncertainty which often improves the performance during inference.

Attention module

Attention modules were first introduced by Vaswani et al., (2017) as a new model architecture called "Transformers" specifically creates to address Natural Language Processing (NLP) problem. The ripple effects of this newly proposed method also reached GAN frameworks. The DCGAN architecture used in all the frameworks studied in this research are very effective at capturing local features. However, the network tends to overlook the interrelationship between long-range structures if it is outside its Receptive Field[26]. Self-Attention Generative Adversarial Networks (SAGAN) was the first model proposed within GAN framework that added an attention module to guide the discovery of features at distant portions of the image (Zhang et al., 2019). Similarly, we can leverage the information in our skip-connected autoencoder component of our generator model to provide the attention guide. By including an attention module in our generator network, we hope to teach the SkipGANomaly model to

[26] Receptive Field (RF) is defined as the size of the region in the input that produces the feature (Araujo et al., 2019).

recognise the distant patterns in the image and thus learn the regions corresponding to anomaly-free samples. For example, by understanding the background noise we might start to decipher the more relevant foreground regions of the image. The notion of the hypothesis with the attention module within the GAN framework is to find the structures within the car image that don't attend well to each other, leading to discrepancies that might lead to us decoding anomalous regions of the arbitrary image during inference. However, attention mechanisms are not easy to train, as the training process tends to be slower due to models being bigger and the inference requiring more computational resources.

6.5 Reflection on the Project

This project provided me with my first experience working with Generative Adversarial Networks, allowing me to explore an entirely new subfield of Computer Vision, and expanding my knowledge of cutting-edge AI technologies. Although the project began based on pre-existing work by Akcay et al., (2018, 2019), I believe the new variants of the frameworks will be able to contribute to the scientific community of Computer Vision technologies in the future. As an extension of the introduced masked SkipGANomaly framework that solves image completion problem, I am motivated to do further research in this field and explore other applications such as adversarial learning via attention modules for anomaly detection.

GLOSSARY

Machine-learning: an academic field that focuses on programs and systems concerning training a model from input data to make predictions from never seen before data drawn from the same distribution as the one used to train the model.

Computer Vision: is a field of artificial intelligence (AI) enabling computers to derive information from images, videos and other inputs, enabling the automation task of human visual systems.

Artificial Neural Networks (ANN): a stack of simple learning algorithms (referred to as layers), where each layer accepts an input and some parameters to gain a more nuanced understanding of the input and generate a series of outputs in a sequential manner.

GitHub: An internet hosting service with version control using Git, allowing user(s) to distribute code and collaborate on projects, as well as the option to revert to previous versions of the project.

PyTorch: an end-to-end open-source machine learning framework, originally developed by Facebook (https://pytorch.org/).

TensorFlow: Similar to PyTorch, a large-scale open-source, distributed machine learning platform for building ML and AI models, originally developed by Google (https://www.tensorflow.org/).

Keras: As an interface for the Tensorflow library, it provides a Python machine-learning API for artificial neural networks (https://keras.io/).

Unsupervised machine learning: training the model to find patterns without their corresponding labels.

Supervised machine learning: training a model from features and their corresponding labels.

Backpropagation: The algorithm that implements gradient descent in neural networks.

Gradient Descent: A mathematical technique to minimize loss. Gradient descent iteratively adjusts weights and biases, gradually finding the best combination to minimize loss.

Sigmoid function: Activation function where the output of the classifier has more than one right answer. It outputs the probability distribution of n possible outcomes in the range of 0 to 1 or -1 to 1, where it looks at each raw output value separately.

Softmax function: Activation function outputting the normalized probability distribution proportional to the exponentials of the input in the range of 0 to 1, where all the elements always sum to one by design, since the outputs are interrelated.

Batch Size: The collection of examples utilised in one training cycle. The batch size determines how many instances comprise a batch.

Epoch: A full training pass through the entire training set, processing each example once. An epoch is equivalent to n / Batch size in a training cycle, where ‘n’ is the total number of examples.

L_1-distance: also known as Manhattan distance, is the sum of the magnitudes of the vectors in a space, or in other words the sum of absolute difference of vector components.

L_2-distance: also known as Euclidean norm, is the shortest distance between two points.

Mean Squared Error (MSE): Average loss calculated using L_2 distance.

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

where:

- n is the number of examples
- y is the actual value of the label
- $\hat{y}$ is the model's predictions for y

Mean Absolute Error (MAE): Average loss calculated using L_1 distance.

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|$$

where:

1 n is the number of examples

2 y is the actual value of the label

3 $\hat{y}$ is the model's predictions for y

Binary Cross-Entropy (BCE): a generalisation of log loss for multi-class classification problems, quantifying the difference between two probability distributions.

Most of the terms in this glossary were adapted from Google's Machine Learning Glossary (https://developers.google.com/machine-learning/glossary).

REFERENCES

Akcay, S. et al. (2018) *GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training*. [online]. Available from: http://arxiv.org/abs/1805.06725 (Accessed 6 December 2022).

Akçay, S. et al. (2019) *Skip-GANomaly: Skip Connected and Adversarially Trained Encoder-Decoder Anomaly Detection*. [online]. Available from: http://arxiv.org/abs/1901.08954 (Accessed 6 January 2023).

Anon (2016) *RI Seminar: Yann LeCun : The Next Frontier in AI: Unsupervised Learning*. [online]. Available from: https://www.youtube.com/watch?v=IbjF5VjniVE (Accessed 17 November 2022).

Anon (2022) *Scopus database* [online]. Available from: https://www.scopus.com/ (Accessed 17 November 2022).

Araujo, A. et al. (2019) Computing Receptive Fields of Convolutional Neural Networks. *Distill*. [Online] 4 (11), e21.

Arjovsky, M. et al. (2017) *Wasserstein GAN*. [online]. Available from: http://arxiv.org/abs/1701.07875 (Accessed 30 November 2022).

Arjovsky, M. & Bottou, L. (2017) *Towards Principled Methods for Training Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1701.04862 (Accessed 30 November 2022).

Bergmann, P. et al. (2019) 'Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders', in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. [Online]. 2019 pp. 372–380. [online]. Available from: http://arxiv.org/abs/1807.02011 (Accessed 9 January 2023).

Brownlee, J. (2019) How to Identify and Diagnose GAN Failure Modes. MachineLearningMastery.com [online]. Available from: https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/ (Accessed 1 December 2022).

Cao, Y.-J. et al. (2019) Recent Advances of Generative Adversarial Networks in Computer Vision. *IEEE Access*. [Online] 714985–15006.

Chandola, V. et al. (2009) Anomaly detection: A survey. *ACM Computing Surveys*. [Online] 41 (3), 15:1-15:58.

Collin, A.-S. & De Vleeschouwer, C. (2020) *Improved anomaly detection by training an autoencoder with skip connections on images corrupted with Stain-shaped noise*. [online]. Available from: http://arxiv.org/abs/2008.12977 (Accessed 8 January 2023).

Donahue, J. et al. (2017) *Adversarial Feature Learning*. [online]. Available from: http://arxiv.org/abs/1605.09782 (Accessed 4 January 2023).

Dziugaite, G. K. et al. (2015) *Training generative neural networks via Maximum Mean Discrepancy optimization*. [online]. Available from: http://arxiv.org/abs/1505.03906 (Accessed 7 January 2023).

Gal, Y. & Ghahramani, Z. (2016) *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. [online]. Available from: http://arxiv.org/abs/1506.02142 (Accessed 10 January 2023).

Goodfellow, I. et al. (2016) *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press.

Goodfellow, I. (2017) *NIPS 2016 Tutorial: Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1701.00160 (Accessed 24 November 2022).

Goodfellow, I. J. et al. (2014) *Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1406.2661 (Accessed 16 November 2022).

Gu, J. et al. (2017) *Recent Advances in Convolutional Neural Networks*. [online]. Available from: http://arxiv.org/abs/1512.07108 (Accessed 23 November 2022).

Gulrajani, I. et al. (2017) *Improved Training of Wasserstein GANs*. [online]. Available from: http://arxiv.org/abs/1704.00028 (Accessed 7 January 2023).

Haselmann, M. et al. (2018) *Anomaly Detection using Deep Learning based Image Completion*. [online]. Available from: http://arxiv.org/abs/1811.06861 (Accessed 10 January 2023).

He, K. et al. (2015) *Deep Residual Learning for Image Recognition*. [online]. Available from: http://arxiv.org/abs/1512.03385 (Accessed 7 January 2023).

Heusel, M. et al. (2018) *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. [online]. Available from: http://arxiv.org/abs/1706.08500 (Accessed 5 January 2023).

Hong, Y. et al. (2019) How Generative Adversarial Networks and Their Variants Work: An Overview. *ACM Computing Surveys*. [Online] 52 (1), 10:1-10:43.

Ioffe, S. & Szegedy, C. (2015) *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. [online]. Available from: http://arxiv.org/abs/1502.03167 (Accessed 28 November 2022).

Isola, P. et al. (2018) *Image-to-Image Translation with Conditional Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1611.07004 (Accessed 4 January 2023).

Karnewar, A. & Wang, O. (2020) *MSG-GAN: Multi-Scale Gradients for Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1903.06048 (Accessed 1 December 2022).

Karras, T. et al. (2019) 'A Style-Based Generator Architecture for Generative Adversarial Networks', in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [Online]. June 2019 pp. 4396–4405.

Karras, T. et al. (2017) *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. [online]. Available from: http://arxiv.org/abs/1710.10196 (Accessed 22 November 2022).

Kingma, D. P. & Ba, J. (2017) *Adam: A Method for Stochastic Optimization*. [online]. Available from: http://arxiv.org/abs/1412.6980 (Accessed 1 December 2022).

Krizhevsky, A. et al. (2012) 'ImageNet Classification with Deep Convolutional Neural Networks', in *Advances in Neural Information Processing Systems*. 2012 Curran Associates, Inc. p. [online].

Available from: https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (Accessed 23 November 2022).

LeCun, Y. et al. (1989) 'Handwritten Digit Recognition with a Back-Propagation Network', in *Advances in Neural Information Processing Systems*. 1989 Morgan-Kaufmann. p. [online]. Available from: https://papers.nips.cc/paper/1989/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html (Accessed 23 November 2022).

Liu, M.-Y. & Tuzel, O. (2016) *Coupled Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1606.07536 (Accessed 22 November 2022).

Metz, L. et al. (2017) *Unrolled Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1611.02163 (Accessed 1 December 2022).

Miyato, T. et al. (2018) *Spectral Normalization for Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1802.05957 (Accessed 5 January 2023).

Murphy, K. P. (2012) *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning series. Cambridge, MA: MIT Press.

Nair, V. & Hinton, G. E. (2010) 'Rectified linear units improve restricted boltzmann machines', in *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. 21 June 2010 Madison, WI, USA: Omnipress. pp. 807–814.

Qin, X. et al. (2020) U^2-Net: Going Deeper with Nested U-Structure for Salient Object Detection. *Pattern Recognition*. [Online] 106107404.

Radford, A. et al. (2015) *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1511.06434 (Accessed 22 November 2022).

Ronneberger, O. et al. (2015) *U-Net: Convolutional Networks for Biomedical Image Segmentation*. [online]. Available from: http://arxiv.org/abs/1505.04597 (Accessed 6 January 2023).

Roth, K. et al. (2017) *Stabilizing Training of Generative Adversarial Networks through Regularization*. [online]. Available from: http://arxiv.org/abs/1705.09367 (Accessed 1 December 2022).

Salimans, T. et al. (2016) *Improved Techniques for Training GANs*. [online]. Available from: http://arxiv.org/abs/1606.03498 (Accessed 1 December 2022).

Schlegl, T. et al. (2017) *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. [online]. Available from: http://arxiv.org/abs/1703.05921 (Accessed 2 December 2022).

Springenberg, J. T. et al. (2015) *Striving for Simplicity: The All Convolutional Net*. [online]. Available from: http://arxiv.org/abs/1412.6806 (Accessed 28 November 2022).

Srivastava, N. et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 15 (56), 1929–1958.

Szegedy, C. et al. (2015) *Rethinking the Inception Architecture for Computer Vision*. [online]. Available from: http://arxiv.org/abs/1512.00567 (Accessed 8 January 2023).

Tabian, I. et al. (2019) A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. *Sensors*. [Online] 19 (22), 4933.

Vaswani, A. et al. (2017) *Attention Is All You Need*. [online]. Available from: http://arxiv.org/abs/1706.03762 (Accessed 11 January 2023).

Xia, X. et al. (2022) GAN-based anomaly detection: A review. *Neurocomputing*. [Online] 493497–535.

Zenati, H. et al. (2019) *Efficient GAN-Based Anomaly Detection*. [online]. Available from: http://arxiv.org/abs/1802.06222 (Accessed 4 January 2023).

Zhang, H. et al. (2019) *Self-Attention Generative Adversarial Networks*. [online]. Available from: http://arxiv.org/abs/1805.08318 (Accessed 11 January 2023).

Zhao, H. et al. (2018) *Loss Functions for Neural Networks for Image Processing*. [online]. Available from: http://arxiv.org/abs/1511.08861 (Accessed 9 January 2023).

APPENDIX A

All the work products of this study are available in GitHub repository.

https://github.com/elnaramammadova/Car_Damage_Detection_Via_Unsupervised_Generative_Adversarial_Networks

Folder structure:

1. /Nugen_dataset:
   - .ipynb jupyter notebook file containing all the exploratory analysis and pre-processing steps performed prior to loading into the models
   - .npy files containing all the filenames for training and test set respectively
   - .csv file containing the project metadata information such as the S3 bucket key, and the size of the individual images.
2. /Method_1_original_GANomaly
3. /Method_2_modified_GANomaly
4. /Method_3_SN_GANomaly
5. /Method_4_Original_SkipGANomaly
6. /Method_5_Masked_SkipGANomaly

Each Method folders contain:

- .ipynb jupyter notebook file containing all the code that was formulated from scratch using TensorFlow and Keras libraries.
- .png files for the generator and discriminator model architectures
- .csv trainlog file containing all the running values for individual losses during training
- /…_gen_imgs folder containing generated images per epoch