Towards Explainable Generative Adversarial Networks

by

Xiaozhuo Yu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Following publications have resulted from the work presented in the thesis:

1. Ambareesh Ravi, Xiaozhuo Yu, Iara Santelices, Fakhri Karray, Baris Fidan, "General Frameworks for Anomaly Detection Explainability: A comparative study". (Accepted in) IEEE International Conference on Autonomous Systems, August 2021.
2. Xiaozhuo Yu, Fakhri Karray, "Understanding the GAN Mode Collapse Through The Discriminator Perspective." (Submitted to IEEE International Joint Conference on Neural Networks 2022)
3. Xiaozhuo Yu, Fakhri karray, "Improving Time Series Generation of GANs through Soft Dynamic Time Warping Loss." (Submitted to IEEE International Joint Conference on Neural Networks 2022)

For Paper 1 Co-authors Ambareesh Ravi was responsible for data processing, model architecture, and model training. Iara Santelices was partly responsible for literature review and experiments. I was responsible for parts of the experiment, algorithm implementation and paper writing. Dr. Baris Fidan was partly responsible for reviewing and guidance.

In all the above papers, Dr. Fakhri Karray was responsible for coordination, guidance, suggestions and review.

Abstract

As Generative Adversarial Networks become more and more popular for sample generation, the demand for human interpretable explanations have also skyrocketed. With the rising popularity of Generative Adversarial Networks (GANs) in generating synthetic data, time series are no exception to this trend. In this work, not only we tackle these two open challenges, we also provide a comparison of GAN usages for data augmentation. In the first challenge, our work demonstrates that while explainable frameworks can be used to provide insights into the Discriminator module, the explanations provided are not enough. To provide deeper insights and analysis we visualize and analyze the Discriminator to explain why object classes can be omitted resulting in mode-dropping or mode collapse. We also create a new "Discriminative Score" for each object, and we show that their distribution is correlated with this score. Finally, we performed an experiment to determine whether missing details are a result of the architecture or the dataset. In the case of conditional GAN, we discovered that the embedding space can reveal human interpretable semantics that can be manipulated through Principal Component Analysis directions to fine control the generated sample.

In tackling time series generation, we proposed two novel loss functions *sDTW-p* and *sDTW-m* based on Soft-Dynamic Time Warping that can be used to improve the generated time series without modifications to the existing architecture. We also present the first evaluation of the generated samples across different sequence length. We show empirically that the result of leveraging our loss functions can lead to a 9% improvement according to our metric.

Lastly, our findings in data augmentations revealed that traditional methods for Convolutional classifiers can be used to improve the training and usage of GANs. Normalization using custom mean and std was found to improve the Fréchet inception distance of the generated sample while having GAN generate data augmented version of the samples can help improve the base classifier when compared to using data augmentation on the generated images directly.

Acknowledgements

I would like to thank all the little people who made this thesis possible.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Figures xi

List of Tables xiii

List of Abbreviations xiv

1 Introduction 1

1.1 Problem Definition . . . 1
1.2 Motivation . . . 2
1.3 Scope . . . 2
1.4 Objective . . . 3
1.5 Thesis Organization . . . 3

2 Background and Literature Review 4

2.1 Generative Adversarial Networks . . . 4
2.1.1 GANs . . . 4
2.1.2 Conditional GAN . . . 5
2.1.3 Deep Convolutional GAN . . . 5
2.1.4 PPGAN . . . 7
2.2 Evaluating GAN . . . 8
2.2.1 Inception Score . . . 8

2.2.2 Fréchet Inception Distance . . . 8
2.3 GAN Explainability . . . 8
2.3.1 Explaining deep networks . . . 9
2.3.2 Visualizing GANs . . . 9
2.3.3 Discriminator Analysis . . . 10
2.4 Data Augmentation . . . 10
2.4.1 Image Data Augmentation . . . 10
2.4.2 Image Augmentation using GAN . . . 10
2.4.3 Image Augmentations for GAN Training . . . 11
2.5 Time Series . . . 12
2.5.1 Dynamic Time Warping . . . 12
2.5.2 Soft Dynamic Time Warping . . . 12
2.5.3 Time Series Generation . . . 13

3 Proposed Solution **15**
3.1 Introduction . . . 15
3.2 Explainable Frameworks . . . 15
3.2.1 Local Interpretable Model-agnostic Explanation (LIME) . . . 15
3.2.2 Integrated Gradient . . . 16
3.2.3 DeepLIFT . . . 17
3.2.4 SHapley Additive exPlanations (SHAP) . . . 17
3.2.5 Models . . . 18
3.3 Discriminator Dissection . . . 18
3.3.1 Network Dissection . . . 18
3.3.2 Object Realness . . . 19
3.3.3 Forced Details Generation . . . 20
3.4 Conditional GAN . . . 22
3.5 Data Augmentation with GAN . . . 23

3.5.1 Normalization . . . 23
3.5.2 Image Augmentations . . . 24
3.6 Soft Dynamic Time Loss . . . 25
3.6.1 sDTW-p . . . 26
3.6.2 sDTW-m . . . 26

4 Experiments and Analysis **27**
4.1 Datasets . . . 27
4.1.1 MNIST . . . 27
4.1.2 FashionMNIST . . . 27
4.1.3 CIFAR-10 . . . 27
4.1.4 SVHN . . . 28
4.1.5 LSUN . . . 28
4.1.6 CelebA . . . 28
4.1.7 Google Stock . . . 28
4.1.8 Energy . . . 29
4.2 Explainable Framework Comparison . . . 29
4.2.1 Model Architectures . . . 29
4.2.2 Results . . . 30
4.3 Conditional GAN . . . 43
4.3.1 Model . . . 43
4.3.2 Ablation . . . 43
4.3.3 Vector Arithmetic . . . 44
4.3.4 Interpolation . . . 45
4.3.5 PCA . . . 45
4.4 Discriminator Dissection . . . 47
4.4.1 Network Dissection . . . 47
4.4.2 Object Realness . . . 48

4.4.3 Custom Dataset . . . 49
4.4.4 Discussion . . . 50
4.5 Data Augmentation . . . 52
4.5.1 Normalization . . . 52
4.5.2 Augmentation Comparison . . . 52
4.5.3 Results . . . 53
4.6 Time Series Generation . . . 56
4.6.1 Model . . . 56
4.6.2 Generator . . . 56
4.6.3 Discriminator . . . 57
4.6.4 Training . . . 57
4.6.5 Evaluation . . . 58
4.6.6 Results . . . 59
4.6.7 Discussion . . . 61

5 Conclusion **63**

References **65**

List of Figures

2.1 Example of GAN training architecture. 5
2.2 Conditional label integration [35]. 6
2.3 DCGAN Generator Architecture [39]. 6
2.4 PGGAN Architecture [24] 7
2.5 Network dissection technique [3] 9
2.6 Hierarchy of image data augmentations [47] 11

3.1 Activation regions for a layer of the PGGAN discriminator. 19
3.2 LSUN church object segmentation differences [5] 21
3.3 Web interface given to annotators. Annotators are asked if an image contains human figure(s). 22
3.4 Custom Accuracy of Fashion MNIST 24

4.1 Comparison of the XAI frameworks on MNIST classes 0 to 4 for base model. 31
4.2 Comparison of the XAI frameworks on MNIST classes 5 to 9 for base model. 32
4.3 Comparison of the XAI frameworks on MNIST classes 0 to 4 for model trained with noise. 33
4.4 Comparison of the XAI frameworks on MNIST classes 5 to 9 for model trained with noise. 34
4.5 Comparison of the XAI frameworks on MNIST classes 0 to 4 for conditional model. 35
4.6 Comparison of the XAI frameworks on MNIST classes 5 to 9 for conditional model. 36

4.7 Comparison of the XAI frameworks on FashionMNIST classes 0 to 4 for base model. . . . 37
4.8 Comparison of the XAI frameworks on FashionMNIST classes 5 to 9 for base model. . . . 38
4.9 Comparison of the XAI frameworks on FashionMNIST classes 0 to 4 for model trained with noise. . . . 39
4.10 Comparison of the XAI frameworks on FashionMNIST classes 5 to 9 for model trained with noise. . . . 40
4.11 Comparison of the XAI frameworks on FashionMNIST classes 0 to 4 for conditional model. . . . 41
4.12 Comparison of the XAI frameworks on FashionMNIST classes 5 to 9 for conditional model. . . . 42
4.13 Generated Images with embedding weights set to 0. . . . 44
4.14 Resulting images from vector arithmetic. . . . 44
4.15 Interpolation results for both MNIST and CelebA dataset. . . . 45
4.16 Wep Application for exploring the conditional GAN embedding space. . . . 46
4.17 Before and after generated images from the web application by changing PCA 2. . . . 47
4.18 Corresponding object semantics of each scale layer. . . . 48
4.19 Identified images from MTurks. . . . 50
4.20 Images generated by DCGAN trained on normal church dataset. . . . 50
4.21 Images generated by DCGAN trained on custom dataset. Yellow circles indicate the presence of a human figure(s). . . . 51
4.22 Generator architecture. . . . 57
4.23 Discriminator architecture. . . . 58
4.24 PCA and t-SNE plots of both the generated and real samples. The shown figures are taken from models trained using a sequence length of 4. Starting from right to left are the base model, *sDTW-p* and *sDTW-m*. . . . 60

List of Tables

3.1 Chosen Objects 20
3.2 Object distribution of the LSUN church dataset. 22
3.3 Photometric vs. geometric transformation [52]. 25

4.1 CelebA Classes 43
4.2 Object realness output using the LSUN church dataset. 49
4.3 Mean and Standard Deviation of the different datasets. 52
4.4 Augmentation parameters. Translation uses a range to indicate the amount of shift with respect to the size of the image. Horizontal flip indicates the chance of flipping an image. 53
4.5 Resulting FID score of the different datasets. 54
4.6 Data Augmentation result MNIST. 54
4.7 Data augmentation result SVHN. 55
4.8 Discriminative Score (lower the better) tested on the *stock* dataset. 61
4.9 Discriminative Score (lower the better) tested on the *energy* dataset. . . . 61

List of Abbreviations

cGAN Conditional Adversarial Network 5

CNN Convolutional Neural Networks 5

DCGAN Deep Convolution Generative Adversarial Network 5

DeepLIFT Deep Learning Important FeaTures 17

DTW Dynamic Time Warping 12

FID Fréchet Inception Distance 8

GAN Generative Adversarial Network 1, 4

IG Integrated Gradient 16

IS Inception Score 8

LIME Local Interpretable Model-agnostic Explanation 15

LSTM Long short-term memory 13

PCA Principal Component Analysis 23

PGGAN Progressive Growing of GAN 7

sDTW Soft Dynamic Time Warping 12

SHAP SHapley Additive exPlanations (SHAP) 17

XAI Explainable Artificial Intelligence 15

Chapter 1

Introduction

1.1 Problem Definition

Generative Adversarial Networks (GAN) [16] are one of the most popular generative model in use today. The most popular usage of GANs is for image generation where synthetic images that are very close to the original samples are generated. For instance, GANs have been leveraged in biomedical informatics to generate medical images [28] with great success. Despite their popularity very little studies have been done to explain this black box model. For systems that cannot be well-interpreted their decisions become hard to trust, especially in sectors, such as healthcare or self-driving cars, where moral and fairness issues have naturally arisen. This need for trustworthy, fair, robust, high performing models for real-world applications have led to the revival of the field of eXplainable Artificial Intelligence (XAI) [18]. As GANs become more involved in critical areas the need for explainability substantially increases. However, as previously mentioned XAI for GANs are rare for visualizing and interpreting the outcome of GANs, researchers have mostly focused on the Generator module. Little research has been focused on interpreting GANs using the Discriminator module. Since GANs are composed of two interacting modules (Generator and Discriminator), it is only fair to analyze both modules for explainability purposes. Furthermore, conditional variants of GANs have not been studied and no explainability has been provided on the differences between conditional and unconditional variant.

1.2 Motivation

This research is meant to be complementary to the work done in [3] and [4] where only the Generator was used for explainability purposes. Additionally, the explanation only focused on the unconditional version of GANs while neglecting the conditional variant which are often more useful since they allow for image generation of specific classes. Since GANs are composed of both the Generator and the Discriminator, the sub networks will affect each other. Therefore, to explain the full network behaviour we should take both sub networks into consideration. One strong motivation for studying the Discriminator module is the possibility of improving the generated samples. In [5] visualization of mode collapse where image details are lacking is shown but the question remains on why specific details were missing. Identifying the mistakes of the Generator using the explanations from the Discriminator can help us understand why certain details are omitted and we can work towards generating more diverse and realistic samples. Lastly, this research is very much aligned with sub-theme 8 where the major focus lies in generative models and being able to explain their decisions. GANs are one of the most popular generative models and if we can provide additional insights it will contribute greatly towards sub-theme 8.

The popularity of machine learning has skyrocketed in the last few decades. Deep Learning models in particular are pushing the boundaries of what is possible [41], [9]. As the model parameters become increasingly complex and the number of layers become deeper, the availability of data can become a bottleneck for machine learning solutions in domain specific problems. In the field of medicine, data accessibility can hinder comparisons, reproducibility, and scientific progress [14]. Data accessibility are not the only issue, low amount of data can result in possible identification of individuals through linkage of data from other sources or residual information [13] resulting in violation of privacy.

The issues mentioned above can be resolved through synthetic data. With synthetic data, it could be shared and published without privacy concerns, or even used to augment or enrich similar datasets [14]. Amongst the methods for synthetic data generation Generative Adversarial Networks (GANs) [46] stood out. GANs have been leveraged for the creation of photorealistic images but their uses for time series data have been limited. Despite their rare usages for time series, GANs have demonstrated remarkable results.

1.3 Scope

This research is focused on bridging the gap between existing literature on GAN explainability by including needed examination into the Discriminator module as well as the con-

ditional GAN variant. Additionally this research also provide a comprehensive comparison of data augmentation with GANs. Lastly, the research provides new novel loss functions that can be used to improve the performance of time series generation with GANs.

1.4 Objective

The main objective of this work is to increase the explainability of GANs. The secondary focus is to provide comparisons as well as potential methods for GAN improvements. To achieve the aforementioned objectives, the following contributions are made in this research:

1. Comparison of different popular explainability frameworks with different GAN models.
2. Exploring the embedding space of conditional GANs to understand the impact of class labels.
3. Dissecting the Discriminator module to gain insights into missed objects during generation.
4. Comparing different data augmentation with GAN.
5. Improve time series generation with GAN through new loss function.

1.5 Thesis Organization

This thesis constitutes five chapters. The first chapter introduces and describes the current problem and challenges faced with GAN explainability. This chapter also introduces the main objects and scope for the proposed solutions. Chapter 2 covers a detailed review of existing literature and works that relates to GANs. Chapter 3 deals with the proposed solutions in great details. Chapter 4 discusses the various datasets used as well as the experiments and analysis of the proposed approaches. The final chapter summarizes and concludes the findings of this work.

Chapter 2

Background and Literature Review

2.1 Generative Adversarial Networks

This chapter discusses the progression of explainable research in the field of Generative Adversarial Networks in detail along with popular works that had breakthroughs in the field. This chapter also focuses on the necessary background for both data augmentation and time series generation along with their respective open challenges.

2.1.1 GANs

Generative Adversarial Networks (GAN)s are a new type of deep generative models that have been used to generate realistic samples. The ability of a GAN to generate realistic samples stem from its two modules: Generator and Discriminator. The Generator module takes in noise/latent Z and outputs a synthesized sample. The Discriminator takes in either fake or real samples and discern whether the input was generated or real. The two modules are trained in a zero-sum game according to the equation defined in equation 2.1 where G is the Generator, D is the Discriminator. An architecture of GAN is depicted in Figure 2.1.

$$\begin{aligned} \min_G \max_D V(G,D) = \mathrm{E}_{x\sim P_{real}(x)}[\log(D(x)] \\ + \mathrm{E}_{z\sim P_Z(Z)}[log(1-D(G(Z))] \end{aligned} \tag{2.1}$$

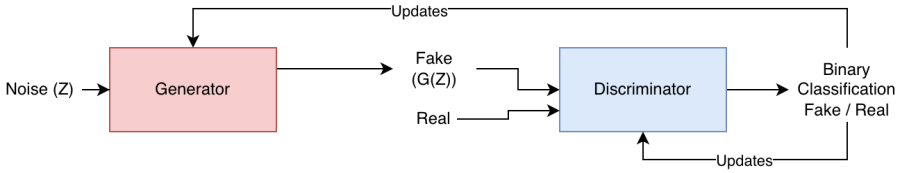

Figure 2.1: Example of GAN training architecture.

2.1.2 Conditional GAN

Although GAN models are capable of generating new random plausible examples for a given dataset, there is no way to control the types of images that are generated other than trying to figure out the complex relationship between the latent space input to the generator and the generated images.

The conditional generative adversarial network, or cGAN for short, is a type of GAN that involves the conditional generation of images by a generator model. Image generation can be conditional on a class label, if available, allowing the targeted generated of images of a given type. To create the conditional version, we add the class label information to the latent z. The Discriminator will also be given the class label information, in doing so, the network can then be trained to generate specific images of a specified class. An example of how conditional label y is integrated into the GAN is given in Figure 2.2.

2.1.3 Deep Convolutional GAN

The Deep Convolutional GAN (DCGAN) architecture was proposed to expand on the internal complexity of the Generator and Discriminator modules. The architecture uses Convolutional Neural Networks (CNN) for the Generator and Discriminator modules rather than multilayer perceptrons. The idea behind DCGAN is to increase the complexity of the Generator network to project the input into a high dimensional tensor and then add deconvolutional layers to go from the projected tensor to an output image. The deconvolutional layers will expand on the spatial dimensions, for example, going from 4×4 x 6 to 8 x 8 $\times$ 1, whereas the convolutional layers in the Discriminator will decrease the spatial dimensions such as going from 64 x 4 x 3 to 32 x 32 x 6. The architecture of DCGAN

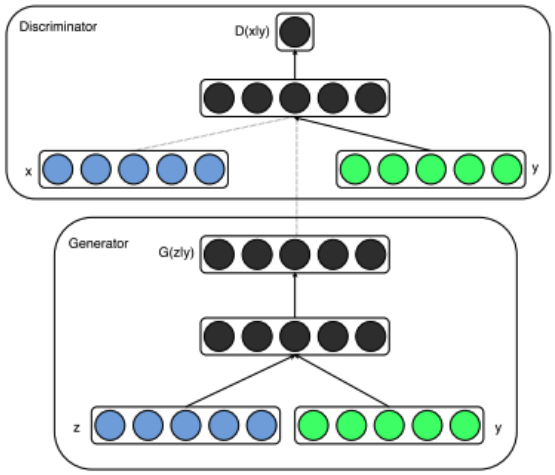

Figure 2.2: Conditional label integration [35].

Generator is depicted in Figure 2.3. The Discriminator architecture is the inverse of the Generator architecture as it performs convolution instead to down sample the image [39].

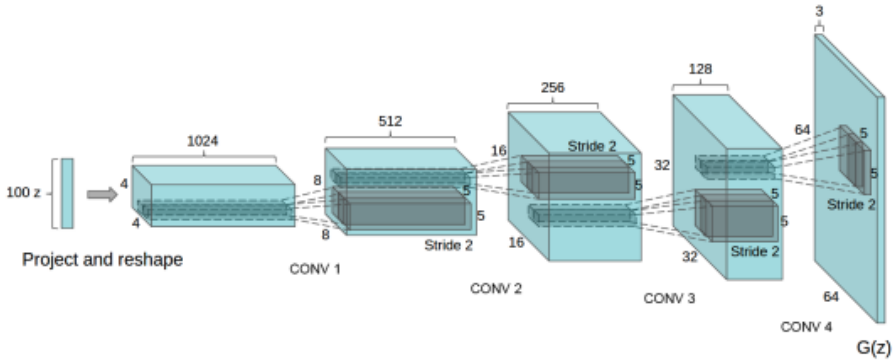

Figure 2.3: DCGAN Generator Architecture [39].

2.1.4 PPGAN

A problem prior to Progressive Growing of GAN (PGGAN) [24] is that GANs have a hard time producing high-resolution images since the Generator must learn to output both large structure and fine details at the same time. The high resolution also makes it easy for the Discriminator to spot missing details and as a result failing the training process. To stabilize the GAN training, the GAN model is progressively grown. Progressively growing the GAN involves using a Generator and Discriminator model with the same general structure and starting with very small images, such as 4×4 pixels. Once training starts, new blocks of convolutional layers are systematically added to both the Generator model and the Discriminator models. The incremental addition of layers allows the sub modules to learn coarse-level details and fine details in the later stages. An example of progressively growing GAN is shown in Figure 2.4.

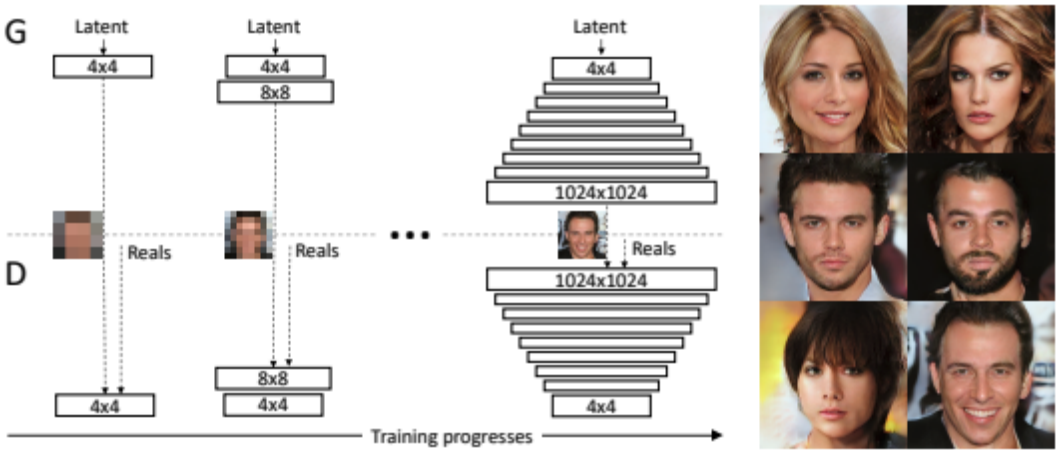

Figure 2.4: PGGAN Architecture [24]

2.2 Evaluating GAN

2.2.1 Inception Score

Inception Score (IS) [45] is one the original methods for measuring the quality of generated samples. In [45], they proposed applying an Inception-v3 [51] network pre-trained on ImageNet [11] to generated samples and then comparing the conditional label distribution with the marginal label distribution. The calculation of IS is depicted in Equation 2.2.

$$IS = exp(\mathbb{E}_{x \sim p_s} D_{KL}(p(y|x)||p(y))) \tag{2.2}$$

Generator that has high IS will be:

1. Capable of generating images with meaningful object which means the condiitonal label distribution $p(y|x)$ has low entropy.
2. Capable of generating diverse images, so that the marginal label distribution $p(y) = \int_x p(y|x)p_g(x)$ has high entropy.

2.2.2 Fréchet Inception Distance

The Fréchet Inception Distance (FID) [20] is an improvement over IS by comparing the statistics of the generated samples to real samples instead evaluating the generated samples in a vacuum. The calculation of FID is shown in Equation 2.3 where Tr represents the trace and $X_r \sim \mathcal{N}(\mu_r, \sum_r)$ and $X_g \sim \mathcal{N}(\mu_g, \sum_g)$ are the 2048 dimensional activations of the Inception v3 pool3 layer for the real and generated samples. A lower FID is correlated with more similar real and generated samples.

$$FID = ||\mu_r - \mu_g||^2 + \text{Tr}(\sum_r + \sum_g - 2\sqrt{(\sum_r \sum_g)}) \tag{2.3}$$

2.3 GAN Explainability

Despite their successes, explainability works for GANs are rare and are often focused on manipulation of the latent vectors. The latent vectors were found to be manipulable; vector arithmetic could be used to generate images that are combinations of two different classes.

2.3.1 Explaining deep networks

One of the first work in explaining Convolutional neural networks (CNN) comes from [49] where salient features are visualized. Another step in improving explainability for CNNs is when techniques such as SHapley Additive exPlanations (SHAP) [31] and Local Interpretable Model-Agnostic Explanations (LIME) [42] came into existence. These techniques focused on scoring saliency to explain the decisions of the networks. Other techniques have also evolved to explain the decisions of networks through the ablation of network units [37].

2.3.2 Visualizing GANs

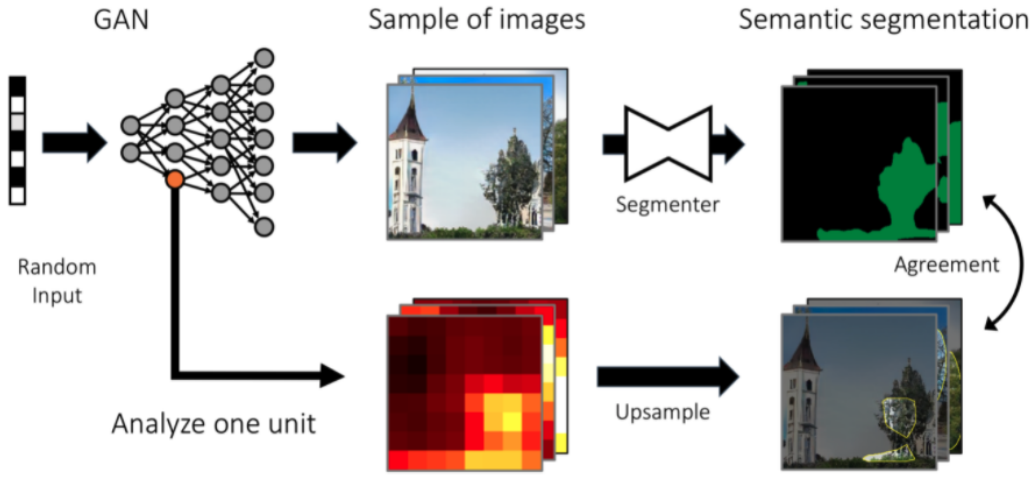

Figure 2.5: Network dissection technique [3]

The most well-known work on visualizing and explaining GANs are from [3] and [4]. In [4] and [3], a technique called *network dissection* is used to identify any units within the network and match those with the same semantic classes. An example of the technique is shown in Figure 2.5. By identifying the units responsible they can then be turned on or off and the effect on image generation can be seen. Another work done by the same authors as [4] is on the visualization of mode collapse. In [5], a semantic segmentation network is used to compare the distribution of segmented objects in the generated images with the target distribution in the training set. Differences in segmentation statistics can then reveal object classes that are omitted by the Generator.

2.3.3 Discriminator Analysis

Literature on studying the Discriminator module is rare and only one relevant paper [29] exists to the author's knowledge. In [29] an enhanced Layer-wise Relevance Propagation (LRP) algorithm called Polarized-LRP is used. The algorithm generates a heatmap-based visualization highlighting the area in the input image that contributes to the network decision. It consists of two parts i.e., a positive contribution heatmap for the images classified as ground truth and a negative contribution heatmap for the ones classified as generated.

2.4 Data Augmentation

2.4.1 Image Data Augmentation

One of the earliest documented case of image augmentation came from AlexNet [27] where data augmentations such as horizontal flip and changing of RGB channels were applied to reduce the error rate of the model. The augmentations employed increased the overall dataset size by a magnitude of 2048 which helped tremendously to reduce overfitting. Since then, a multitude of techniques have been used such as GANs in 2014, Neural Style Transfer [15] and Neural Architecture Search (NAS) [63] in 2017.

A breakdown of different image data augmentations can be found in Figure 2.6.

2.4.2 Image Augmentation using GAN

One of the most popular usages of GAN for image augmentation lies in the medical domain. Within the medical domain, GANs are widely used for medical image synthesis. The reason being that generated images can help to overcome the privacy issues related to diagnostic medical image data and as well as tackle the insufficient number of positive cases of different pathologies [59]. For instance, GANs have been leveraged successfully to generate computed tomography (CT) images of liver lesions. Using a limited dataset of 182 liver lesions GANs were leveraged to synthesize high quality liver lesion ROIs. The classification performance using only classic data augmentation yielded 78.6% sensitivity and 88.4% specificity. By adding the synthetic data augmentation, the results increased to 85.7% sensitivity and 92.4% specificity. Another instance of using GANs for image augmentation is in [55] where it improved the classification of chest X-ray (CXR) from 85% to 95%.

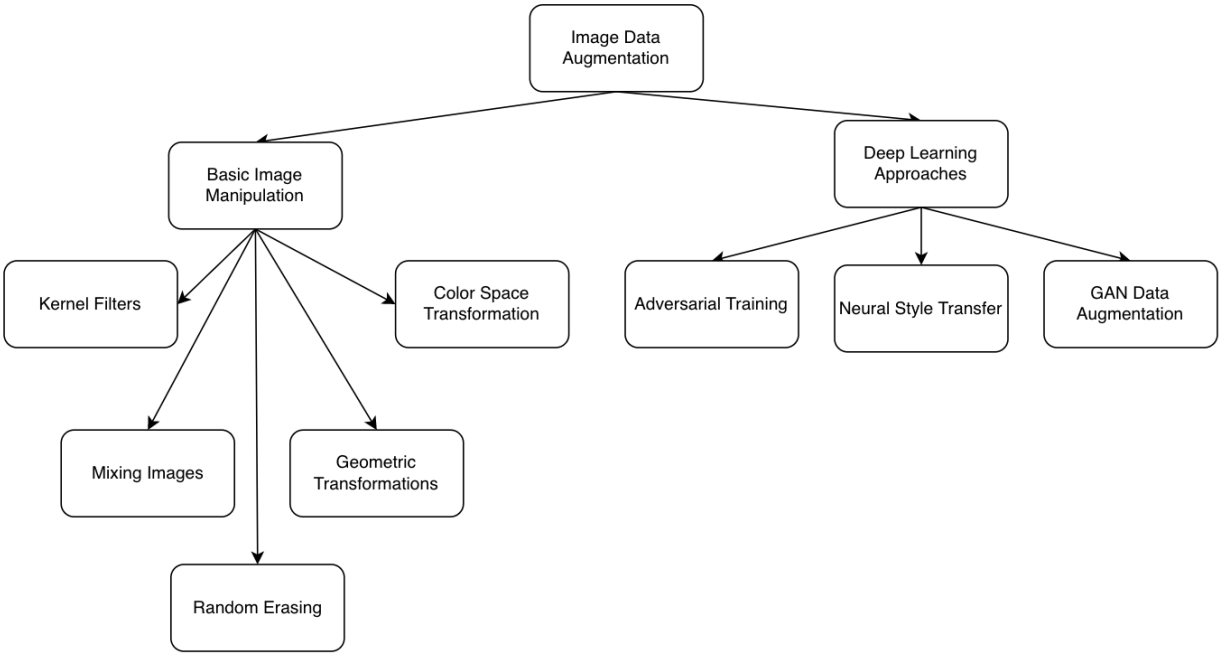

Figure 2.6: Hierarchy of image data augmentations [47]

2.4.3 Image Augmentations for GAN Training

While data augmentations have been widely studied to improve the accuracy and robustness of classifiers, the potential of image augmentation in improving GANs has not been thoroughly investigated [62]. In [62] a systematic study on the effectiveness of various existing augmentation techniques for GAN training was done. From the study, a vanilla GAN can attain generation quality that is on par with recent state-of-the-art results if augmentations are applied on both real and generated images. The result of the generation is measured using FID [21]. It is important to note that the generated images are not augmented in anyway, augmentations are applied to both the real and fake images.

Algorithm 1 Dynamic Time Warping pseudocode.

```
Input: s : array[1..n], t : array[1..m]
Initialize DTW = array[0..n, 0..m]
for i = 0 to n do
  for j = 0 to m do
    DTW[i, j] = ∞
  end for
  DTW[0, 0] = 0
  for i = 1 to n do
    for j = 1 to m do
      cost = d(s[i], t[j])
      minimum = min(DTW[i − 1, j],
                    DTW[i, j − 1],
                    DTW[i − 1, j − 1])
      DTW[i, j] = cost + minimum
    end for
  end for
end for
return DTW[n, m]
```

2.5 Time Series

2.5.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a dynamic programming algorithm that was first introduced for speech recognition [44]. However, it can also be used to calculate the dissimilarity of two time-series. DTW has been leveraged for different time series related tasks such as finding optimal alignments [40] and clustering [23]. The pseudocode for DTW can be found in algorithm 1.

2.5.2 Soft Dynamic Time Warping

Soft Dynamic Time Warping (sDTW) [10] is a reformulation of DTW based on soft-minimum that allows it to be a differentiable loss function. The value and gradient of sDTW can be computed with quadratic time and space complexity. Given two time-series

$X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$, and their cost matrix C of size $n \times m$. The cost $C[i, j]$ is the cost of x_i to y_i and following 1 is computed as:

$$C_{i,j} = f(x_i, y_i) + \min\{C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}\} \tag{2.4}$$

where $f(x_i, y_i)$ is the chosen distance function which in our case is the Euclidean distance (ED). To convert from 2.4 to *sDTW*, the *min* operator is replaced a differentiable soft minimum that is defined as:

$$min^{\gamma>0}\{C_1, .., C_n\} := -\gamma \log \sum_{i=1}^{n} e^{-C_i/\gamma} \tag{2.5}$$

where γ is the controlled smoothness factor. If γ is set to 0 then it would yield the original DTW score. For more details regarding the derivation of *sDTW* see [10].

2.5.3 Time Series Generation

A variety of methods have been used to generate synthetic time data. Starting from autoregressive recurrent networks trained via the maximum likelihood principle [56] to WaveNet, an autoregressive neural network [53] for speech and audio generation. The true breakthrough for synthetic data generation came with the introduction of Generative Adversarial Networks (GANs). When GANs were first introduced they focused mostly on the generation of synthetic images but gradually there has been a shift towards time series generation. The first GAN adapted for time series was C-RNN-GAN [36] that leveraged long short-term memory (LSTM) networks for the Generator and Discriminator to generate data from the previous time step. Following C-RNN-GAN, Recurrent Conditional GAN improved its predecessor by removing the dependence on previous outputs while conditioning on additional inputs [14]. TimeGAN is a new model that combines the unsupervised approach of GANs with the ability to control conditional temporal dynamics from supervised autoregressive models to generate synthetic samples. The model is trained via the supervised loss and trained embedding networks.

Leveraging Soft Dynamic Time Warping (s-DTW) with GANs have been before in [33] for gesture synthesis. In [33], the new loss function based on *s-DTW* replaced the Discriminator. The result of the new model trained with s-DTW ended up generating more realistic samples.

For time series generation leveraging GANs with different loss function only one paper was found where a multivariate dynamic time warping term was added as a penalty term [6]. The authors in [6] used a penalisation coefficient (Dynamic Time Warping) with the standard GAN loss function to improve the generated samples of electrocardiogram (ECG). While this work looks similar there are a few key differences. Since Dynamic Time Warping is not differentiable no gradients will be generated and thus no directions are provided for the networks during training. By leveraging soft-DTW which is a differentiable loss function, gradients can be generated. The datasets used in [6] are all ECG data which are periodic and have a very low feature dimensions. The authors do not explore datasets with higher dimensions and no variation of the sequence lengths were tested. These are all covered in our paper making it more comprehensive. We also leveraged different evaluation metrics to cover more important aspects of generated samples such as *fidelity* and *diversity*.

Chapter 3

Proposed Solution

3.1 Introduction

The existing works on GAN explainability lack insights into the Discriminator and are mostly focused on the Generator. To bridge the gap, several experiments such as applying XAI frameworks to visualize the Discriminator, Discriminator dissection and Object Realness are applied in this chapter. Along with the base GAN, the conditional variant is also explored through the embedding space. Finally, the chapter also examines in details different comparisons of data augmentation leveraging GANs as well as new loss functions to improve time series generation.

3.2 Explainable Frameworks

Most Explainable Artificial Intelligence (XAI) frameworks operate on class predictions and since the GAN Discriminator module can be considered as a binary classifier, these methods can be applied without modifications. Four major XAI methods are explored in this section as well as any modifications that are required.

3.2.1 Local Interpretable Model-agnostic Explanation (LIME)

Local Interpretable Model-agnostic Explanation (LIME) [42] is a generalized explainability framework that utilizes local *surrogate* models to explain the individual classification or

regression decisions of a black box model. The surrogate model is a simplification of the original complex model and only approximates well for a subset of the data. LIME essentially analyses the change in probability scores on multiple instances of a reference input with added noise or change in the value to provide suitable explanations. A set of data samples containing perturbed versions X_p of the reference image x is generated by switching off or replacing the pixels of the interpretable components and the score for each of the samples in the set is calculated. The surrogate model $f_s \epsilon F_s$ is then trained on X_p learning to weight the patches of pixels based on proximity $\pi(x)$ as in equation (3.1) where f_o is the Discriminator module, F_s is the family of surrogate models, $\Omega(f_s)$ is the complexity of f_s, and $\mathcal{L}$ is the Loss function. Finally, the pixels with the largest weights denote the explanation for the reference image which indicates the essential attribute that makes the model decide on that particular class as follows:

$$explanation(x) = \underset{f_s \epsilon F_s}{\operatorname{argmin}} \mathcal{L}(f_o, f_s, \pi_x) + \Omega(f_s) \tag{3.1}$$

3.2.2 Integrated Gradient

Integrated Gradient (IG) [50] computes the gradient of the model's prediction with respect to its input features. IG is built on top of two axioms that were not satisfied by any other attribution methods at the time of its creation. The two axioms are:

1. Sensitivity
2. Implementation Invariance

Sensitivity. An attribution method satisfies *sensitivity* if for every input and baseline that differ in one feature but have different output then the differing feature should be given a non-zero attribution [50].

Implementation Invariance. Two networks are functionally equivalent if their outputs are equal for all inputs, despite having different implementations. An attribution method satisfies implementation invariance if the attributions are always identical for two functionally equivalent networks.

IG can be calculated in five different steps

1. Create baseline which is neutral for the output prediction

2. Create a linear interpolation between the baseline and the input image
3. Calculate gradients to measure the relationship between features changes the changes in the model's prediction
4. Compute the numerical approximation through averaging gradients
5. Ensure attribution vales are accumulated across multiple interpolated images are all in the same units

3.2.3 DeepLIFT

Deep Learning Important FeaTures (DeepLIFT) [48] is a method for decomposing the output prediction of a neural network on a specific input by backpropagating the contributions of all neurons to every feature. DeepLIFT compares the activation of each neuron to its 'reference activation' and assigns contribution scores according to the difference.

DeepLIFT explains the difference in the output from some reference output in terms of their different. Suppose the output of a given input has a difference of δt with the reference output. The contribution scores to the differences of the activations of neurons in any intermediate layer with their reference state can be calculated as:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{3.2}$$

The contribution scores are computed via different rules such as Linear rule, Rescale rule or RevealCancelRule. For more detailed explanation regarding the individual rules please see [48]. In our case the contributions are calculated using the Rescale rule.

3.2.4 SHapley Additive exPlanations (SHAP)

SHapley Additive exPlanations (SHAP) [31] is a feature attribution based explainability method that measures the importance of an input feature concerning the output prediction. SHAP is an additive feature attribution method that uses *Shapely values*, a concept from coalitional game theory that describes how fairly the prediction is distributed among the input features which in our case signifies the quantification of the contribution of each input feature towards the final prediction. Due to this property of shapely values, SHAP provides consistent global interpretation for each data sample. SHAP replaces features with

random variables to determine its contribution towards the final output prediction through the relative difference from the original prediction. The weights for DeepLIFTSHAP $\pi_z()$ can be determined by the equation (3.3) where $|z'|$ is the number of features considered for the coalition and M is the maximum coalition among features.

$$\pi_z(z') = (M-1)/((M/|z'|) \times |z'| \times (M-|z'|)) \tag{3.3}$$

3.2.5 Models

The different XAI frameworks are also applied to different models to examine whether the frameworks themselves can reveal the differences in architecture. We use three different models, a base model, a model trained with Gaussian Noise and the conditional version. The model trained with Gaussian Noise is meant to act as a weight penalty which has been shown to prevent the Discriminator from overfitting thus improving the quality of the Generated sample [17].

3.3 Discriminator Dissection

3.3.1 Network Dissection

In [2] a general framework called Network Dissection is proposed to quantify the interpretability of Convolution Neural Networks through the evaluation of alignment between individual units and a set of human interpretable semantics. In Network Dissection, each unit u of a CNN layer computes an activation function $au(x,p)$ that outputs a signal at every image position p given a test image x. The probability of an event is true for all position p and images x is given by $\mathbb{P}_{x,p}[\cdot]$. Given the probability a threshold t_u can be defined as $t_u \equiv max_t\, \mathbb{P}_{x,p}[a_u(x,p) > t] \geq 0.01$. Visualization can be viewed by highlighting the activation region $p|a_u(x,p) > t_u$ that is above the threshold. An example of activation highlight for the Discriminator is shown in Figure 3.1.

To correlate the activation region with semantic concepts, a measure of agreement between each filter and a visual concept c is done using a segmentation model [58] as $S_c : (x,p) -> 0,1$. The agreement between concept c and unit u is calculated using the intersection over union (IoU) ratio:

$$IoU_{u,c} = \frac{\mathbb{P}_{x,p}[S_c(x,p) \wedge (a_u(x,p) > t_u]}{\mathbb{P}_{x,p}[S_c(x,p) \vee (a_u(x,p) > t_u]} \tag{3.4}$$

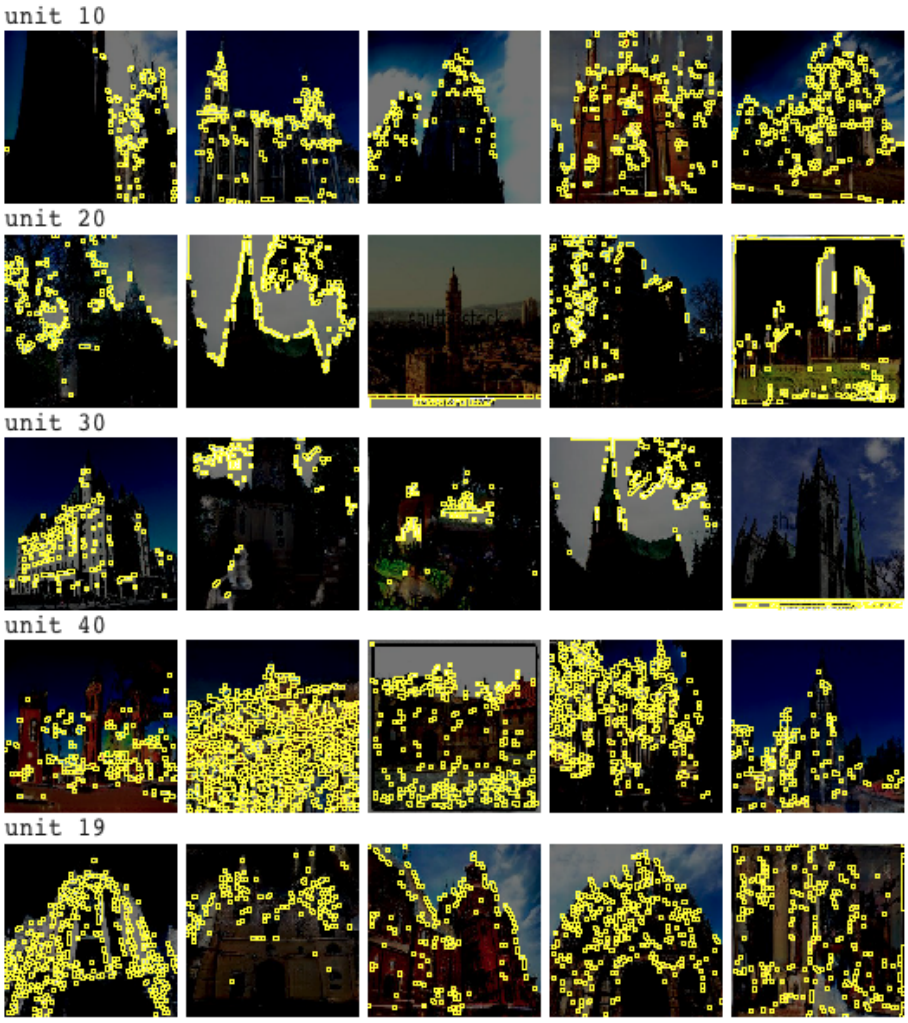

Figure 3.1: Activation regions for a layer of the PGGAN discriminator.

3.3.2 Object Realness

An initial hypothesis regarding object realness is that the degree of realness of the different object classes should be correlated to their presence in the generated samples. To be specific, if we apply image segmentation on the generated images and obtain a distribution of the segmented object classes, they should be correlated with the degree of realness

obtained from the Discriminator. An example of object class distribution of the LSUN church dataset can be seen in Figure 3.2 where the object distribution from the generated sample has very large delta against the training distribution which indicates an omission of details. The object distribution that we see in Figure 3.2 should be correlated with the individual object class's degree of realness. The correlations make sense intuitively since GANs are trained in a zero-sum game and the Generator attempts to trick the Discriminator by focusing on generating samples that are more "real" to the Discriminator to prevent loss.

The individual object segments can be obtained using the Unified Perceptual Parsing network [58] which is trained to detect over 300 different object classes. Once the segments are obtained we apply Algorithm 2 for a specific image to obtain the realness of segmented objects. Rather than calculating the realness of all possible classes we focused on a subset of them. The chosen subsets of classes are shown in 3.1 and is the same chosen objects used in 2. The degree of realness for a chosen object is then averaged across the entire dataset.

Table 3.1: Chosen Objects

LSUN Church
Building
Tree
Road
Grass
Person
Earth
Sidewalk
Mountain

3.3.3 Forced Details Generation

In Table 3.2 we can see that there are discrepancies with the distribution of objects generated and the real dataset. To determine whether object details are missing because of model architecture or the dataset we decided to create a new dataset that is composed of more details, specifically we created a new dataset from LSUN church where each image contained one or more person. To do so we leveraged Amazon's Mechanical Turk (MTurk).

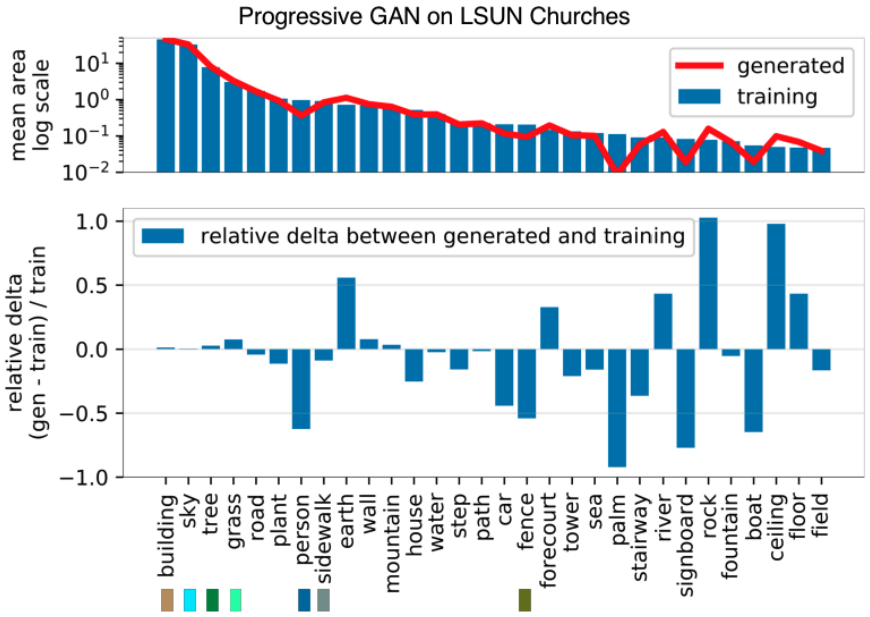

Figure 3.2: LSUN church object segmentation differences [5]

Algorithm 2 Object Realness Calculation

Input: image x_i
Initialize $desiredObjects = \{...\}, outputs = \{\}$
$objects = getObjectsFromImage(x_i)$
for *object* **in** *objects* **do**
  if $objectindesiredObjects$ **then**
    $blank = createBlankImage()$
    $before = discriminator(blank)$
    $merged = mergeImage(blank, object)$
    $after = discriminator(merged)$
    $outputs[object] = before/after$
  end if
end for

Using MTurk, we asked each user whether an image contained a person. An example of the web app on MTurk is shown in Figure 3.3.

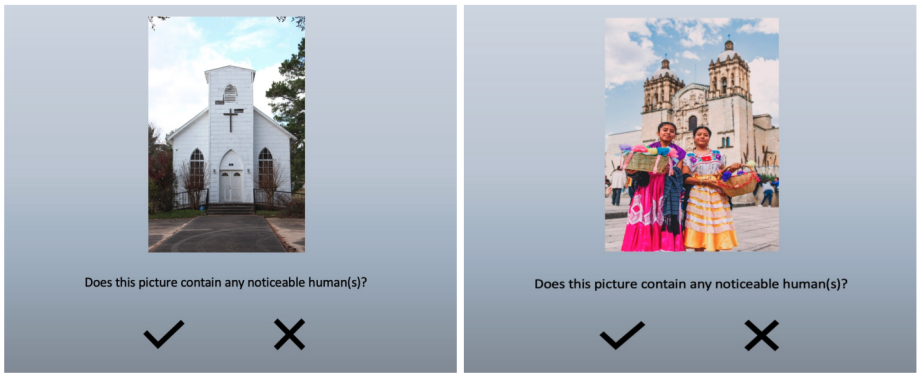

Figure 3.3: Web interface given to annotators. Annotators are asked if an image contains human figure(s).

Table 3.2: Object distribution of the LSUN church dataset.

Object	**% in real**	**% in generated**
Building	98.9	90
Tree	70.5	80
Road	41	20
Grass	37	40
Person	12.7	0
Earth	29.7	30
Sidewalk	24	20
Mountain	12.6	20

3.4 Conditional GAN

In this section, we provide insights into the embedding space of conditional GANs via vector arithmetic and interpolation. These qualities are also present in the latent vector of unconditional GANs. Our aim here is to understand whether the same qualities can be extended to the conditional embedding space. To determine the effect of the conditional labels we set the embedding weight of the Generator to be zeroes and examine the resulting generated images. In doing so it gives us insights into the embedding space as well as how

the Generator leverages the class information for image generation. In [34] for evaluating learned representation of words, the work showed arithmetic operations could reveal linear structure in the representation space. The most famous example demonstrated that vector("King") - vector("Man") + vector("Woman") resulted in a vector whose nearest neighbor was the vector for Queen. A similar finding for the unconditional GAN exists where similar arithmetic on the Z vectors was performed. Experiments working on only single samples per concept were unstable but averaging the latent Z vector for three different examples showed consistent and stable generations that semantically obeyed the arithmetic. They also have shown that face pose is also modeled linearly in Z space [39]. We perform a similar experiment as above but instead of working with the latent vector we work directly with the embedding weights. Specifically, we perform vector arithmetic by adding the embedding weights of different classes and observing the output. In [39] the latent vector Z show cased interpolation capabilities, to extend it to the conditional variant we perform a similar a technique but for the embedding space. To replicate the interpolation in [39] we create a linear interpolation of the different embedding weights and observe the generated outputs. As the different embedding weights revealed linear interpolation, we apply Principal component analysis (PCA) [7] to the embedding weights to see if the PCA direction can reveal any semantics that are easily understandable.

3.5 Data Augmentation with GAN

The goal of this section is to firstly examine whether a custom standarization of image pixel values can lead to better performance in the generated images i.e. lower or higher FID score. The second part of the section is focused on providing comprehensive comparisons of different image augmentations involving GAN to see whether generated augmented images can improve the base classifier accuracy when compared to traditional image augmentation and augmentation involving non-augmented synthetic images.

3.5.1 Normalization

General normalization for GANs is 0.5 for the mean and the standard deviation to change the input range of images to [-1, 1]. The reason behind this range is that the Generator module contains a hyperbolic tanh activation at the end which works in the [-1, 1] range. However, normalization of the dataset according to the mean and standard deviation of the dataset can improve the classification performance. For instance, we trained a ResNet50 [19] on the Fashion-MNIST [57] and the resulting accuracy on the test set is captured in

Figure 3.4. While the difference was slight, the test accuracy of the model that was trained with a custom normalization performed better.

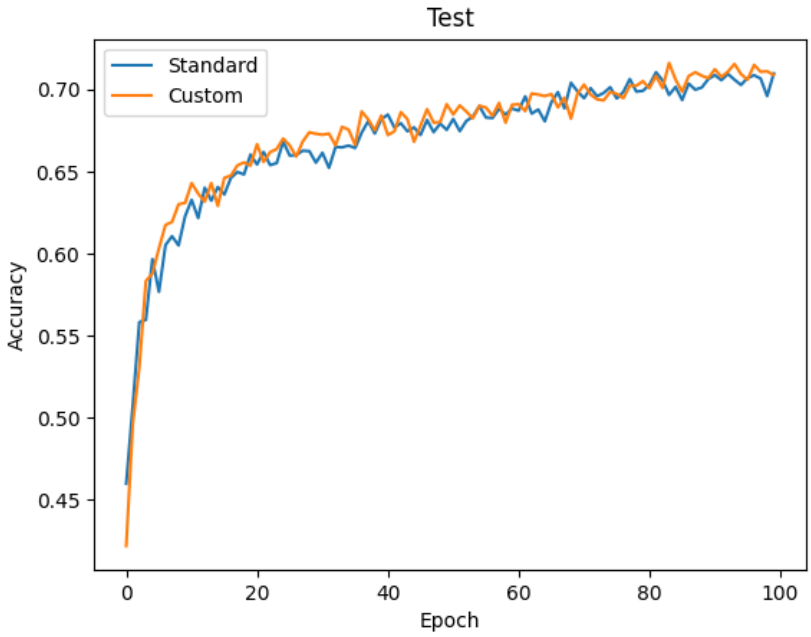

Figure 3.4: Custom Accuracy of Fashion MNIST

3.5.2 Image Augmentations

For data augmentation, instead of focusing on all possible image augmentation techniques we focused on geometric transformations only. From Table 3.3 we can see that the most improvement came from geometric transformations, other augmentations such as PCA and color space transformations were worse hence why we chose to focus on geometric transformations. While random erasing performed the best, we avoided random erasing as it is a non-label preserving transformation.

Within the geometric transformations, we focus on the following: rotation, translation, flipping and scaling.

Table 3.3: Photometric vs. geometric transformation [52].

Augmentation	Top-1 Accuracy	Top-5 Accuracy
Baseline	$48.13 \pm 0.42\%$	$64.50 \pm 0.65\%$
Flipping	$49.73 \pm 1.13\%$	$67.36 \pm 1.38\%$
Rotating	$50.80 \pm 0.63\%$	$69.41 \pm 0.48\%$
Cropping	$61.95 \pm 1.01\%$	$79.10 \pm 0.80\%$
Color Jittering	$49.57 \pm 0.52\%$	$67.18 \pm 0.42\%$
Edge Enhancement	$49.29 \pm 1.16\%$	$66.49 \pm 0.84\%$
Fancy PCA	$49.41 \pm 0.84\%$	$67.54 \pm 1.01\%$

Rotation. Rotation augmentations are done by rotating the image left or right based on the desired degree. The degree can range between 1 and 359. Slight rotations can be beneficial but on datasets such as MNIST when the rotation increases the label of the data is no longer preserved.

Translation. In translation, images are shifted left, right, up, or down which can be useful to avoid positional bias. Translation depending on shift can be non-label preserving when features are shifted out of the image.

Scale. Scale involves randomly scaling the image within the specified interval. Depending on the scale used it can be a non-label preserving technique.

Flipping. Flipping is separated into horizontal or vertical flip. Horizontal axis flipping however, is much more common and depending on the dataset flipping can be non-label preserving. For instance, flipping data from MNIST can be non-label preserving as a 6 or could become a 9.

3.6 Soft Dynamic Time Loss

In this section we explore the novel loss functions used in our experiment. By leveraging *sDTW* we create two new loss functions *sDTW-m* and *sDTW-p* for the Generator module that are used to improve the baseline performance.

3.6.1 sDTW-p

For a batch training data X_b and the corresponding generated data G_b we compute the $sDTW$ loss ($sLoss$ for short) of the generated data and the real batch as $sLoss = sDTW(X_b, G_b)$. The loss of the Generator for a given batch is shown in equation 3.5 where Z_b is a batch of latent input. The intuition behind $sDTW$-p is that by minimizing the differences between the generated sample and the real sample, it should allow for more realistic samples to be generated.

$$\min_G \log(1 - D(G(Z_b))) \tag{3.5}$$

To create $sDTW - p$ we add the $sLoss$ for the batch to the existing generator loss in equation 3.5, the resulting loss for a batch of training data is then shown in equation 3.6.

$$\min_G \log(1 - D(G(Z_b))) + sDTW(X_b, G_b) \tag{3.6}$$

3.6.2 sDTW-m

The $sDTW - M$ loss works in the opposite direction of $sDTW - p$, rather than adding the $sLoss$ we subtract it, this results in equation 3.7. While we wish to minimize the differences between generated and real samples, diversity is still important. By subtracting the resulting $sLoss$ we are forcing the optimizer to optimize by generating more realistic samples as well as increasing the distance between generated samples and real samples. In situations where the data is simple and non-complex it can be easy for the Generator to memorize data. By forcing it to increase the data it makes it harder for the Generator to memorize the data and thus higher quality is generated.

$$\min_G \log(1 - D(G(Z_b))) - sDTW(X_b, G_b) \tag{3.7}$$

Chapter 4

Experiments and Analysis

In our experiments, since GANs tend to be difficult to train and different datasets have different image sizes each experiment will have its own set of models.

4.1 Datasets

4.1.1 MNIST

The MNIST [12] database of handwritten digits is composed of a training set of 60,000 examples, and a test set of 10,000 examples. Each image is 28 x 28 and each of the written digits are centered by computing the center of mass of the pixels and translated.

4.1.2 FashionMNIST

Fashion-MNIST [57] is a dataset of article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from ten classes. It is meant to be a drop-in replacement for MNIST, but it is much more difficult to achieve the same level of accuracy.

4.1.3 CIFAR-10

The CIFAR-10 [26] dataset consists of 60000 32x32 colour images in ten classes, with 6000 images per class. There are 50000 training images and 10000 test images. The ten classes

in the dataset are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

4.1.4 SVHN

SVHN [38] is a real-world image dataset that is created from house numbers in Google Street View images. The dataset contains 10 classes, 1 for each digit and there are 73257 digits for training and 26032 digits for testing. Each image is 32 x 32 and centered around a single character.

4.1.5 LSUN

LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop [61] dataset contains millions of color images for scenes and objects. The labels for this dataset are available based on human's effort for labeling in conjunction with several different image classification models. The images are from parent databases Pascal Voc 2012 and 10 million Images for 10 Scene Categories. The total number of images in the dataset is over 59 million and is split into 10 scene categories and 20 object categories. Scene categories include bedroom, bridge, church outdoor, classroom, conference room, dining room, kitchen, living room, restaurant, tower. The 20 object categories include: airplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, tv-monitor. For our purpose we used the church outdoor scene category which is composed of 126,227 images.

4.1.6 CelebA

CelebFaces Attributes Dataset (CelebA) [30] is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 10,177 number of identities, 202,599 number of face images, and 5 landmark locations, 40 binary attributes annotations per image.

4.1.7 Google Stock

The stock dataset [60] is composed of daily historical Google stocks data from 2004 to 2019. For each day the volume, high, low, opening, closing and adjusted closing prices

are included forming 6 features. The dataset is continuous, aperiodic and the features are correlated with each other. The dataset has a total of 3686 entries.

4.1.8 Energy

The energy dataset [8] contains information regarding house temperature, humidity conditions as well as the appliance energy data. The dataset is consisted of 19735 samples with each sample containing 28 features. For more details regarding the individual features please refer to [8].

4.2 Explainable Framework Comparison

4.2.1 Model Architectures

Base Model

The base Generator module takes in latent size of 100 uses the latent information to generate 28 x 28 images. Each layer of the Generator block is composed of one ConvTranspose2d, one Batch Norm (BN) [22] and a ReLU [1] activation. A hyperbolic tan activation is used in the last layer to normalize the output into the -1 to 1 range. The BN layer is used as it is known to improve the performance, speed of learning and to alleviate vanishing and exploding gradients ensuring smooth propagation of the gradients to the early layers in CNNs. The ReLU activation layer introduces non-linearity by retaining the positive values while clipping the negative values to zero. The base Discriminator module is a mirror of the Generator module but instead of using the latent as the input it receives a 28 x 28 image. The ConvTranspose2d is replaced by normal convolutional layers and ReLU is replaced by leaky ReLU [32]. The last layer of the Discriminator is the sigmoid layer which give us whether the image is real or fake. Leaky ReLU layer is used with 0.2 leakiness which alleviate the dying ReLU problem by returning low-weighted, negative values instead of clipping negative values to zero. In [39] ReLU for the generator creates a bounded activation which allows the model to learn more quickly and saturate the color space of the training distribution. The leaky ReLU worked well for Discriminator especially for higher resolution modeling. These are the reasoning behind why the Discriminator and Generator activations differ. The models were all trained using the Adam optimizer [25] with 0.2 for the alpha value.

Noise Model

The noise model has the same architecture as the base model the only difference is that each Discriminator block has a GaussianNoise layer added. Each GaussianNoise layer perturbs the input to prevent the Discriminator from overfitting.

Conditional Model

The conditional model shares a similar architecture with the base model, but both the Generator and the Discriminator takes in the conditional label.

4.2.2 Results

The four different frameworks: SHAP, LIME, IG, DeepLIFT and their ability to provide insights are shown in Figure 4.1 to 4.12. The aim of each framework is to capture the positive and negative contribution of the image features to the Discriminator's output. From the figures we can see that DeepLIFT SHAP gave the best result regarding how the Discriminator decides an image is real or fake. Regardless of the input image the SHAP output is clear and precise regarding the contribution of each pixel which is due to its ability to provide global interpretation. The worst of the XAI framework was LIME which produced non-sensical outputs. The inability of LIME to give sensible pixel contributions can be attributed to the fact that the LIME surrogate model is meant to be an approximation of the original model and the performance depends on the degree of approximation. Additionally, all other frameworks allow a baseline to be passed in while LIME does not. The baseline helps to identify remove noise and fine tune the attributions. This meant that LIME tend to be faster when it comes to producing results. Out of the remaining three frameworks, DeepLIFT was able to produce the best results. While IG's results look good for the Fashion MNIST dataset the output for the MNIST dataset was not as good. The reason behind IG's results is that similar to LIME IG uses linear interpolation between the baseline provided and the input image and since the Discriminator is nonlinear IG is unable to provide good approximations.

Comparing the different models, we can see that the conditional model has the most defined output. For instance, for class 9 the conditional model showed that the most important attributes are in oval circle and the tail. These details were not as visible in the other classes. For the base model and the noise model the outputs were very similar, and it is hard to decipher if one model is better than the other.

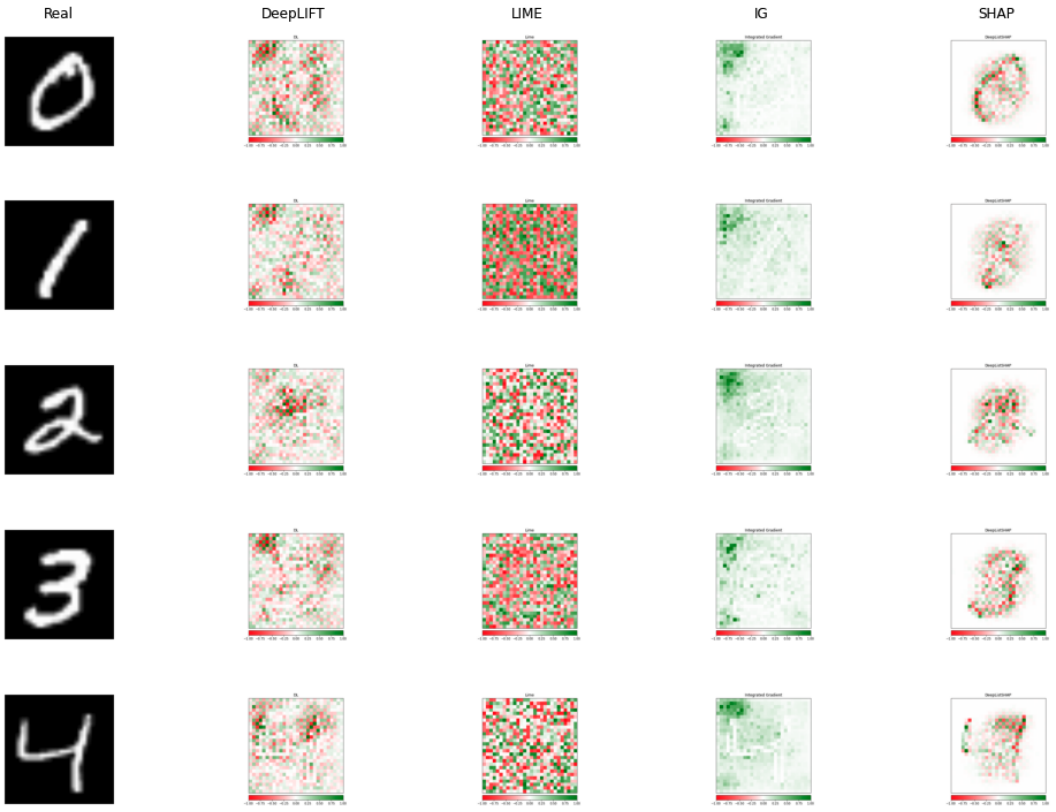

Figure 4.1: Comparison of the XAI frameworks on MNIST classes 0 to 4 for base model.

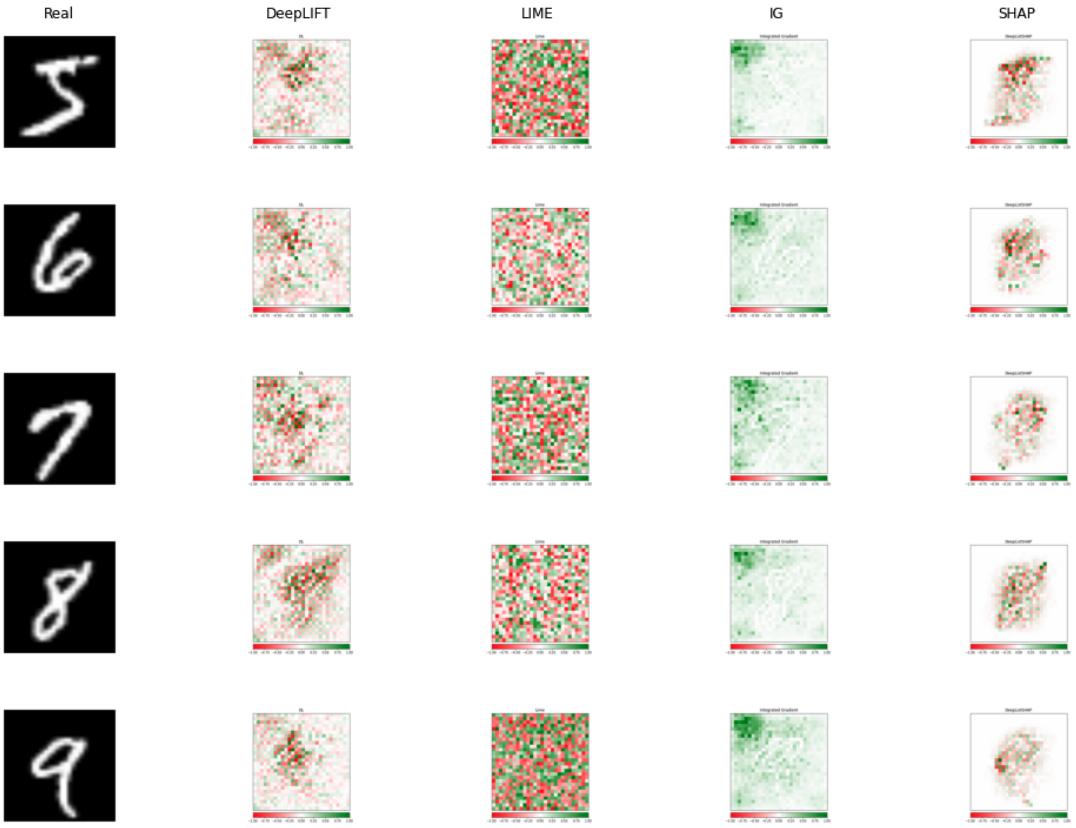

Figure 4.2: Comparison of the XAI frameworks on MNIST classes 5 to 9 for base model.

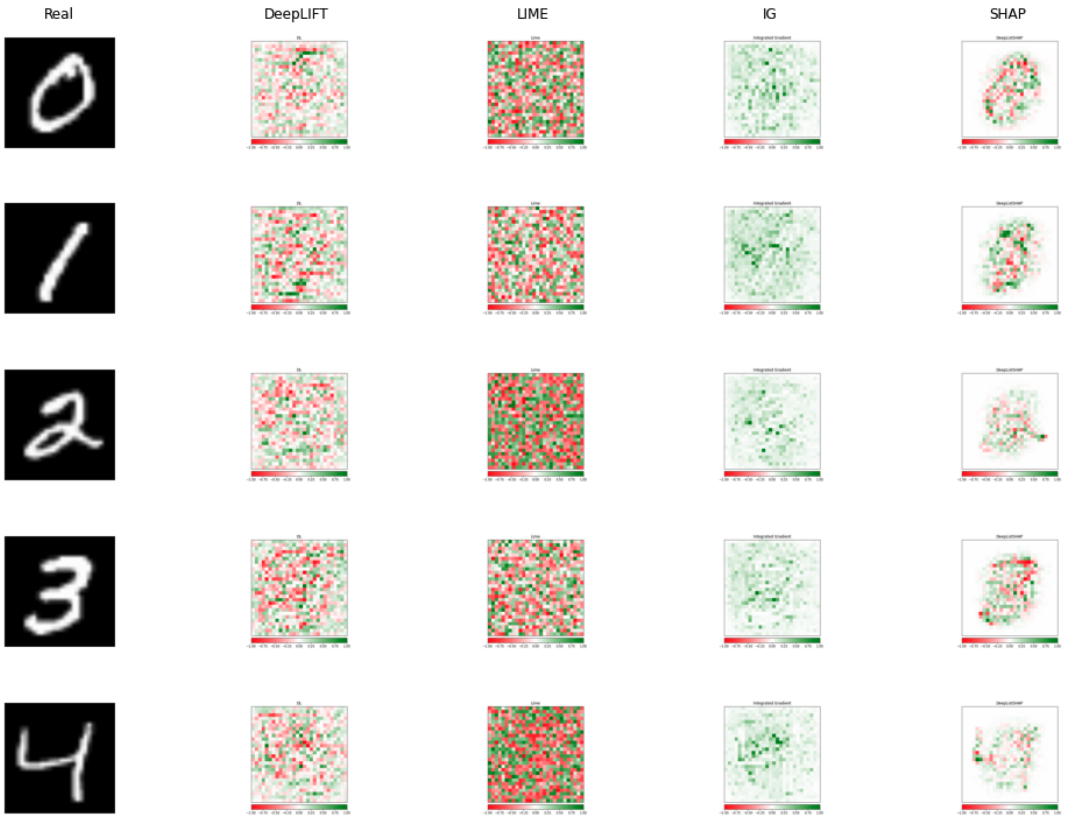

Figure 4.3: Comparison of the XAI frameworks on MNIST classes 0 to 4 for model trained with noise.

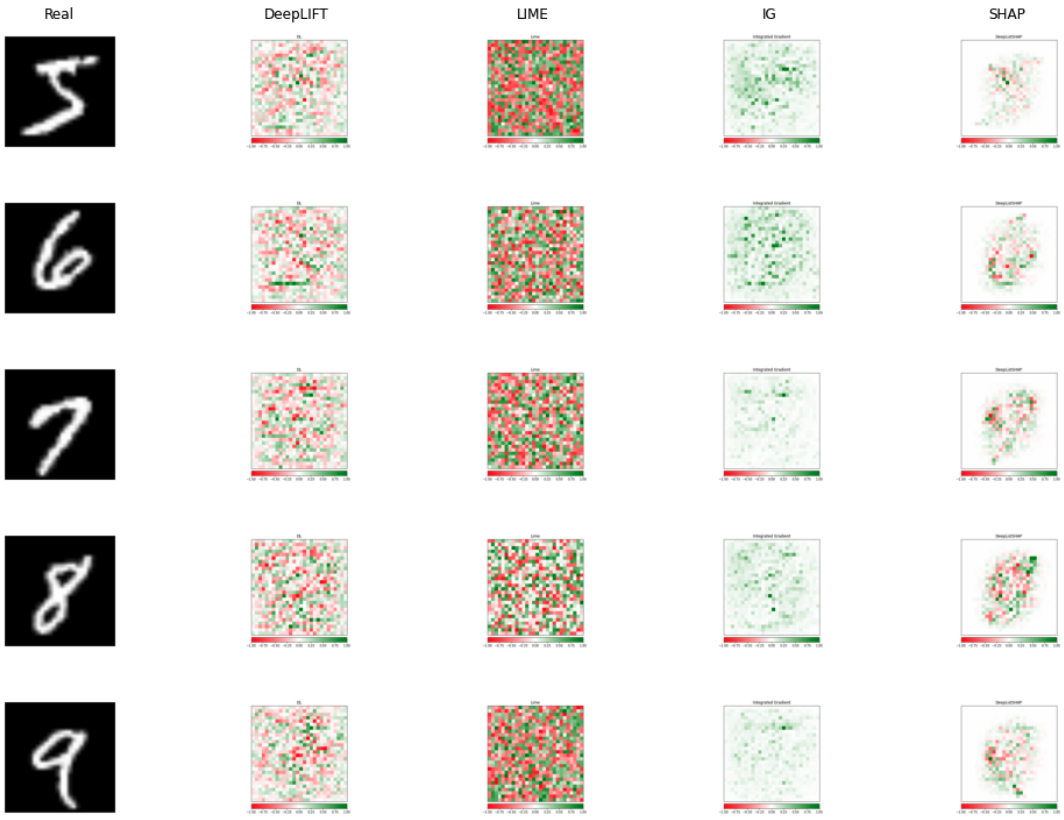

Figure 4.4: Comparison of the XAI frameworks on MNIST classes 5 to 9 for model trained with noise.

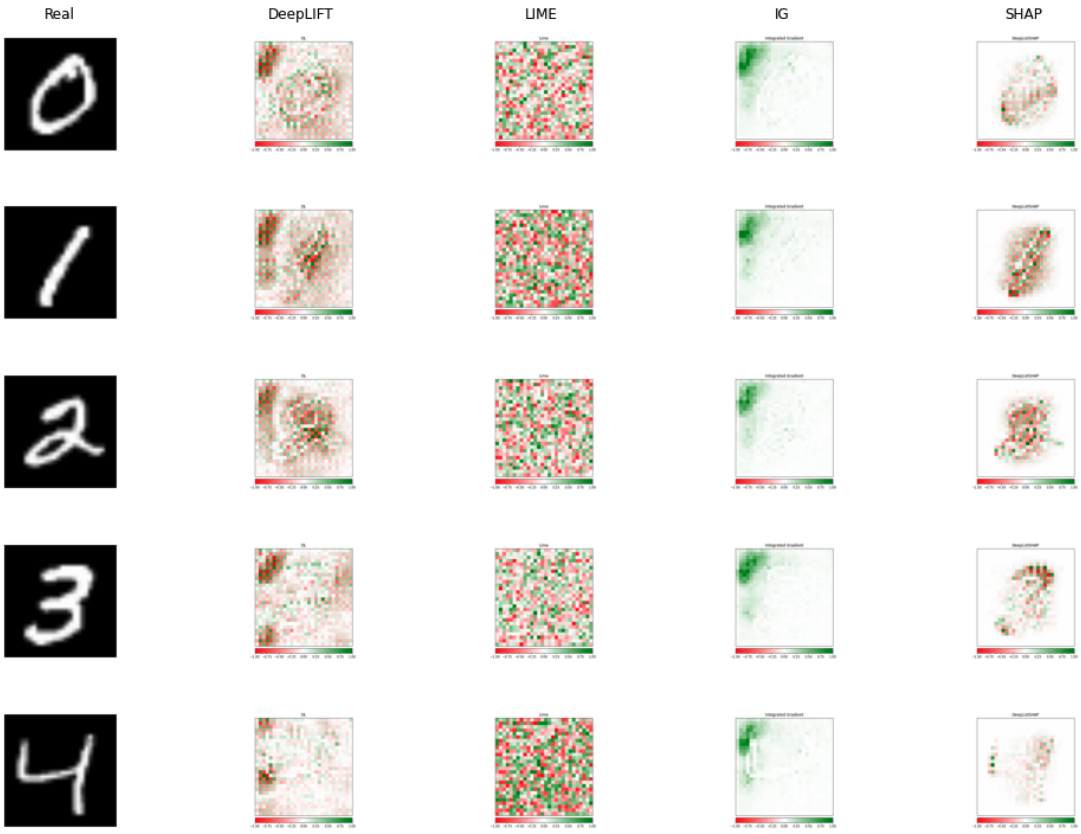

Figure 4.5: Comparison of the XAI frameworks on MNIST classes 0 to 4 for conditional model.

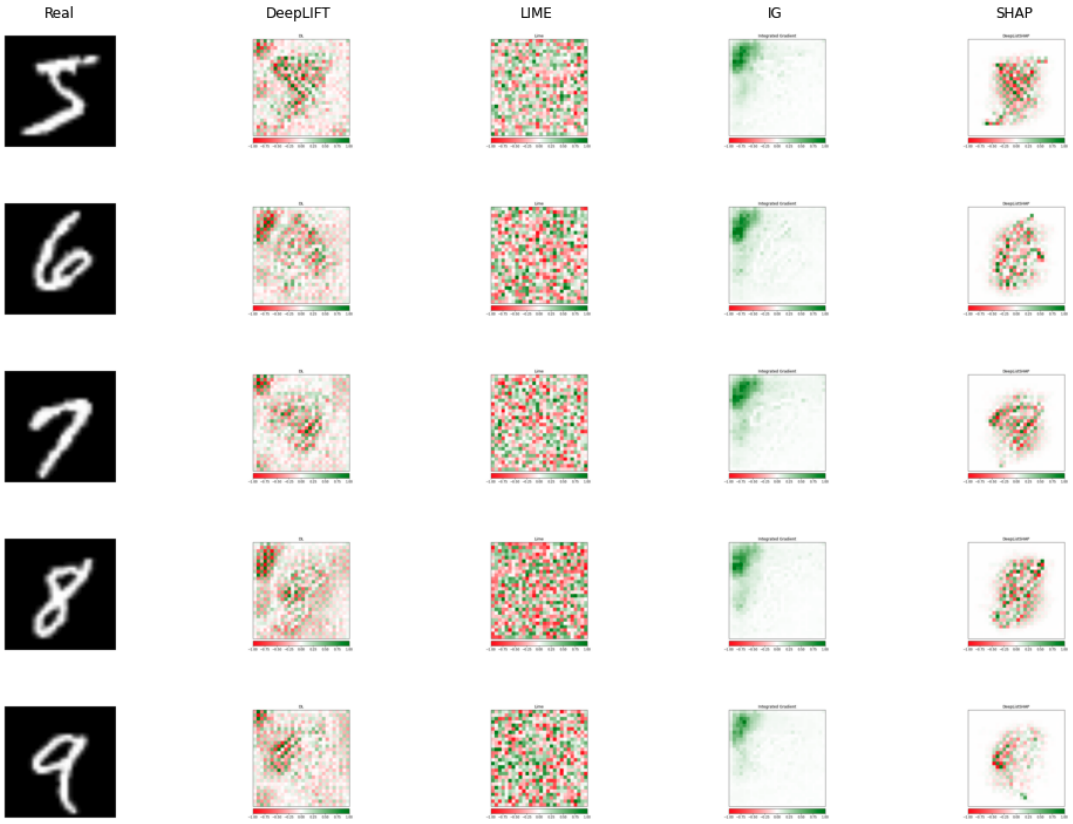

Figure 4.6: Comparison of the XAI frameworks on MNIST classes 5 to 9 for conditional model.

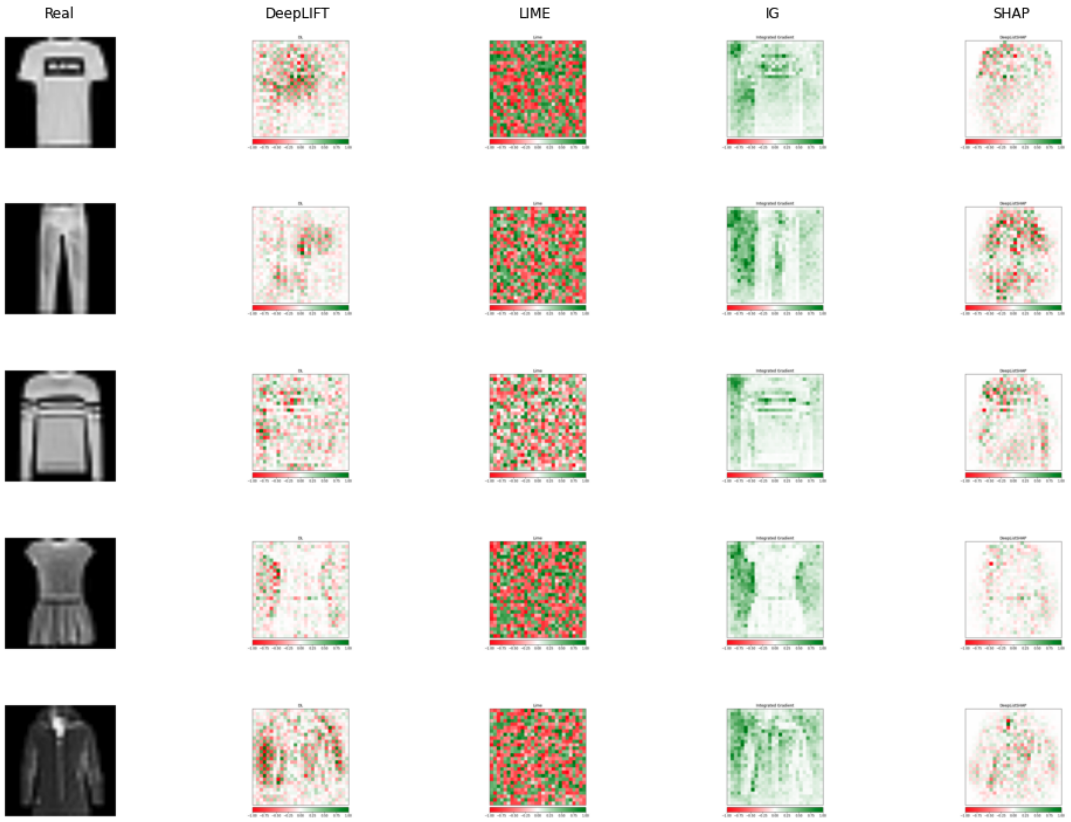

Figure 4.7: Comparison of the XAI frameworks on FashionMNIST classes 0 to 4 for base model.

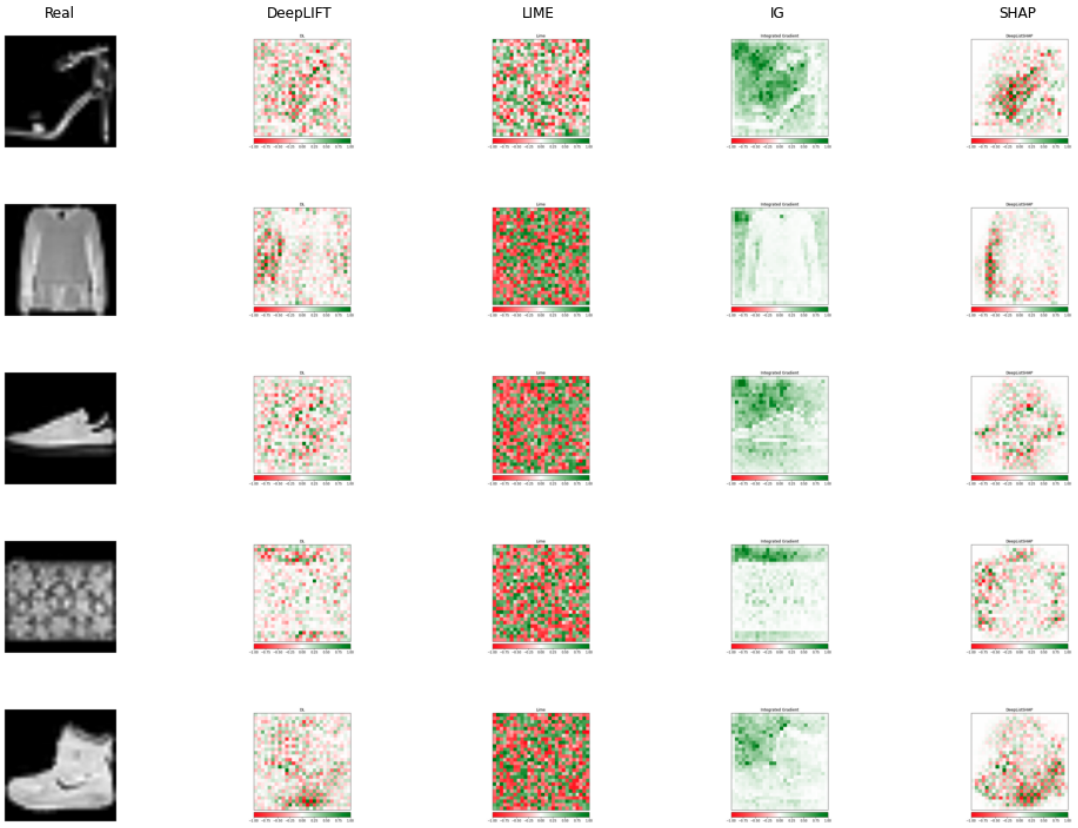

Figure 4.8: Comparison of the XAI frameworks on FashionMNIST classes 5 to 9 for base model.

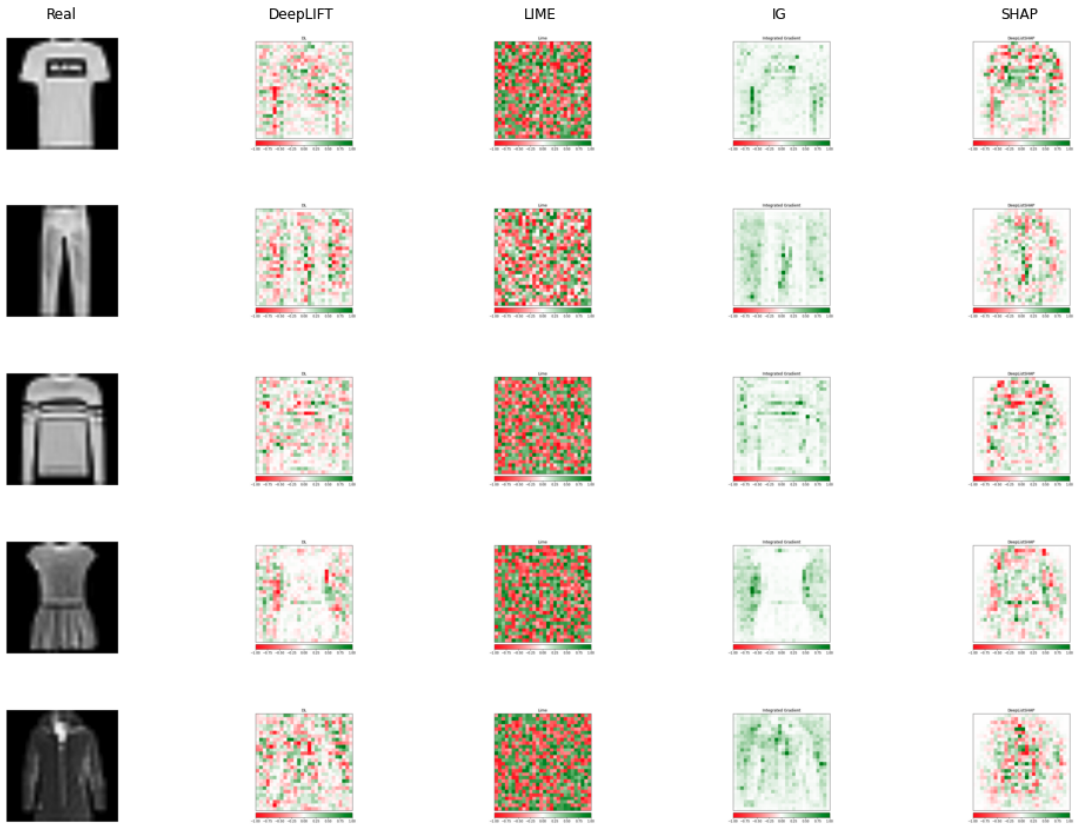

Figure 4.9: Comparison of the XAI frameworks on FashionMNIST classes 0 to 4 for model trained with noise.

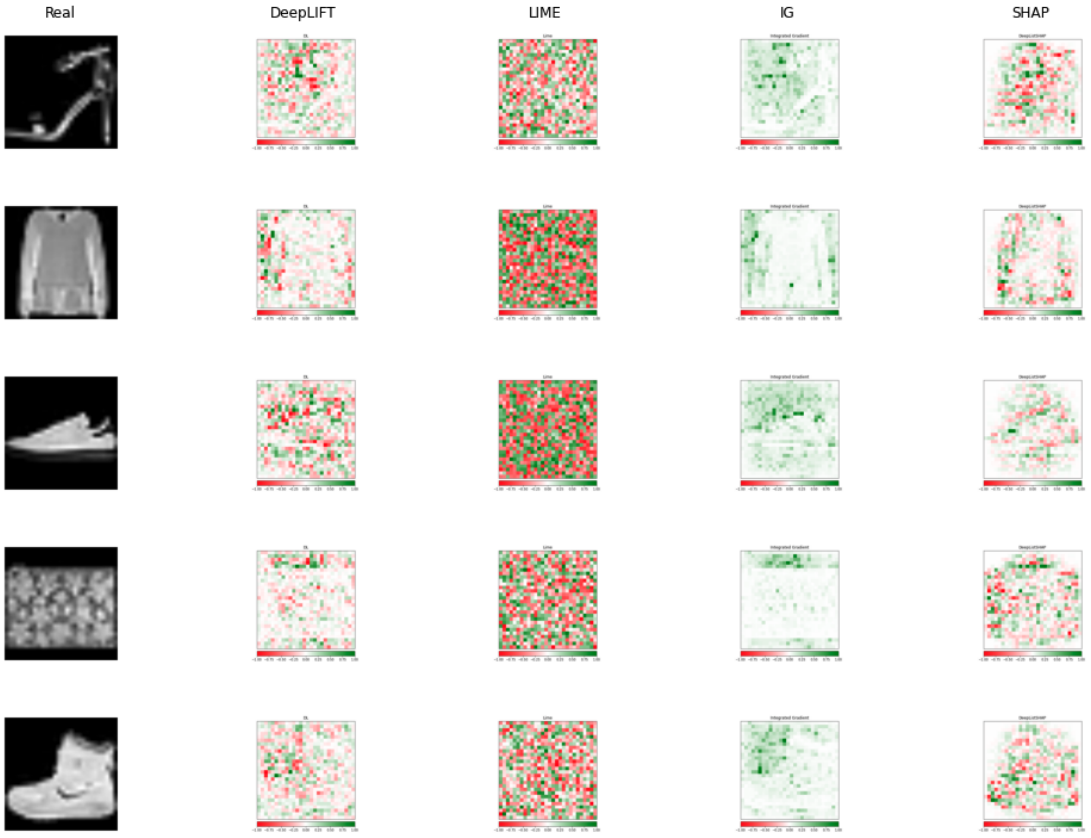

Figure 4.10: Comparison of the XAI frameworks on FashionMNIST classes 5 to 9 for model trained with noise.

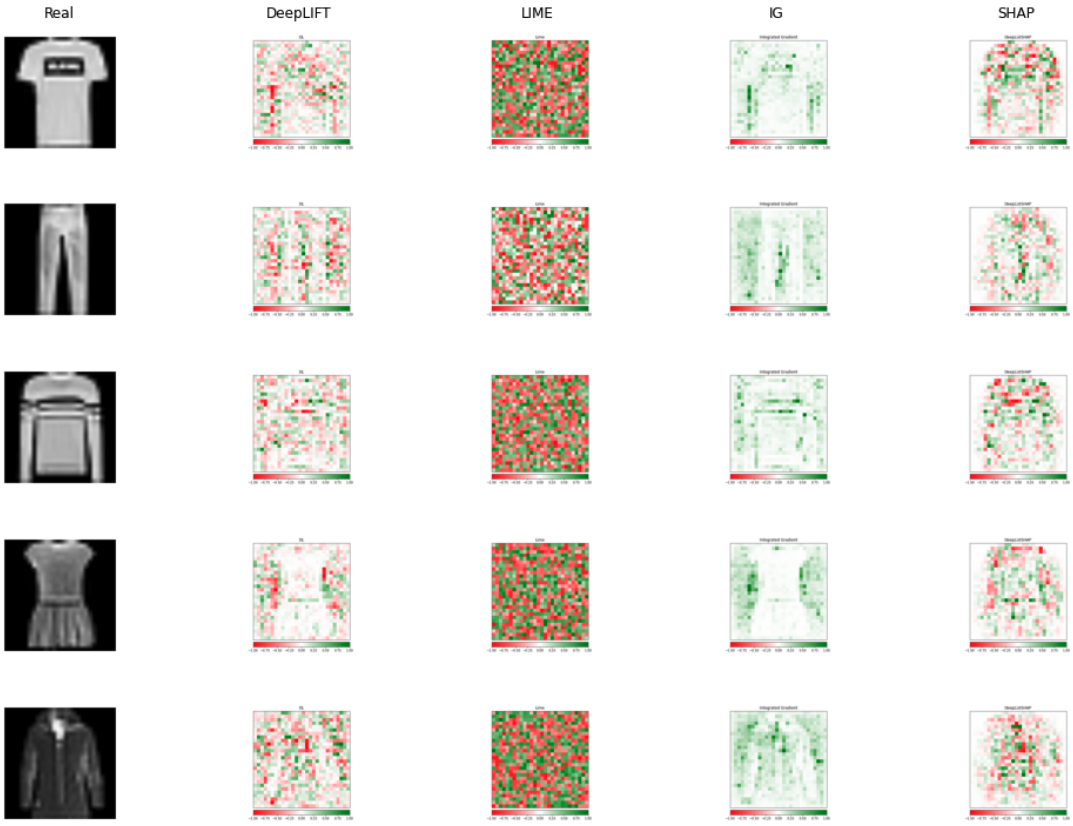

Figure 4.11: Comparison of the XAI frameworks on FashionMNIST classes 0 to 4 for conditional model.

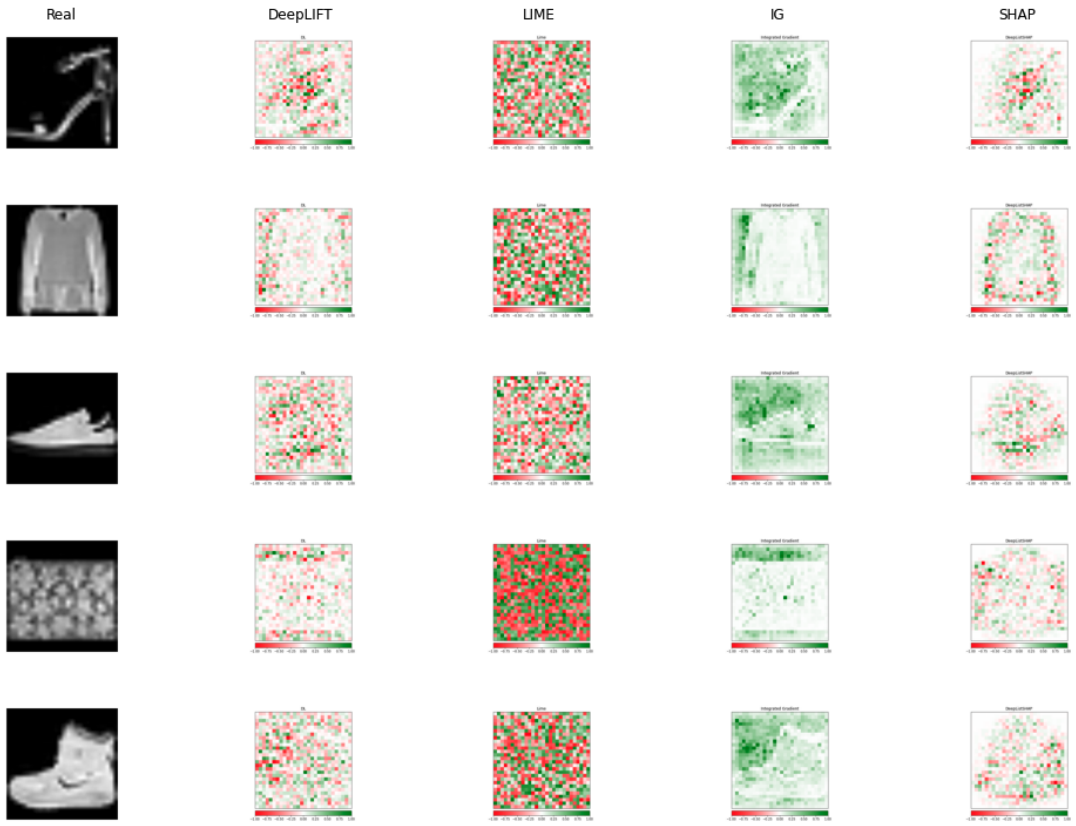

Figure 4.12: Comparison of the XAI frameworks on FashionMNIST classes 5 to 9 for conditional model.

4.3 Conditional GAN

For the conditional GAN exploration, we used the MNIST and CelebA dataset. The CelebA dataset is further processed into 8 different classes which is shown in Table 4.1.

Table 4.1: CelebA Classes

Attributes	**Mapping**
Old Female with glasses	0
Young Female with glasses	1
Old Male with glasses	2
Young Male with glasses	3
Old Female	4
Young Female	5
Old Male	6
Young Male	7

4.3.1 Model

In this experiment we used two different models one for each of the dataset. The MNIST model is designed to generate 28 x 28 images while the CelebA model will generate 64 x 64 images. The MNIST model does not use any convolutional layers and instead rely on Dense layers. The CelebA model is similar to the conditional model in Section 4.2.1 but it has additional layers to generate images at the 64 x 64 scale.

4.3.2 Ablation

From Figure 4.13 we can see that without the embedding weights the Generator is still capable of generating images that resembles human faces but facial features such as eyes, mouths are missing. For the MNIST model we can see something resembling 8 being generated. The result of the figure suggests that the latent vector provides the base of the generated images. The conditional information of labels will add the information needed to fine tune the base image into the desired class.

Figure 4.13: Generated Images with embedding weights set to 0.

4.3.3 Vector Arithmetic

Figure 4.14: Resulting images from vector arithmetic.

In Figure 4.14 we can see that given the same latent information, if we add the embedding weights of different classes together the resulting images is a mixture of the two. For instance, if we add 1 and 7 together, we might expect something that resembles 9 to

be generated. With the CelebA dataset, we can see that by adding young male and young female we get a more feminine face with long hair.

4.3.4 Interpolation

We provide the corresponding linear interpolation in Figure 4.15. As the interpolation points are generated linearly, we can see a linear transformation of the images as they go from 3 to 8. Similarly, we see how the hair become longer and the face is more feminine as we move towards the young female class.

Figure 4.15: Interpolation results for both MNIST and CelebA dataset.

4.3.5 PCA

From the ablation study we can see that the conditional information fine tunes the results from the base latent vector. Additionally, we understand that there potentially exists linear relationship between the different embedding weights. Using PCA we broke down

the embedding weights of the conditional GAN and we discovered that there exists easy human interpretable transformations for each PCA direction. Taking the top 6 PCA revealed the following:

1. PCA 1 is responsible for generating feminine features
2. PCA 2 is responsible for generating glasses
3. PCA 3 is responsible for aging
4. PCA 4 is responsible for masculine features
5. PCA 5 is also responsible for feminine features
6. PCA 6 is also responsible for glasses

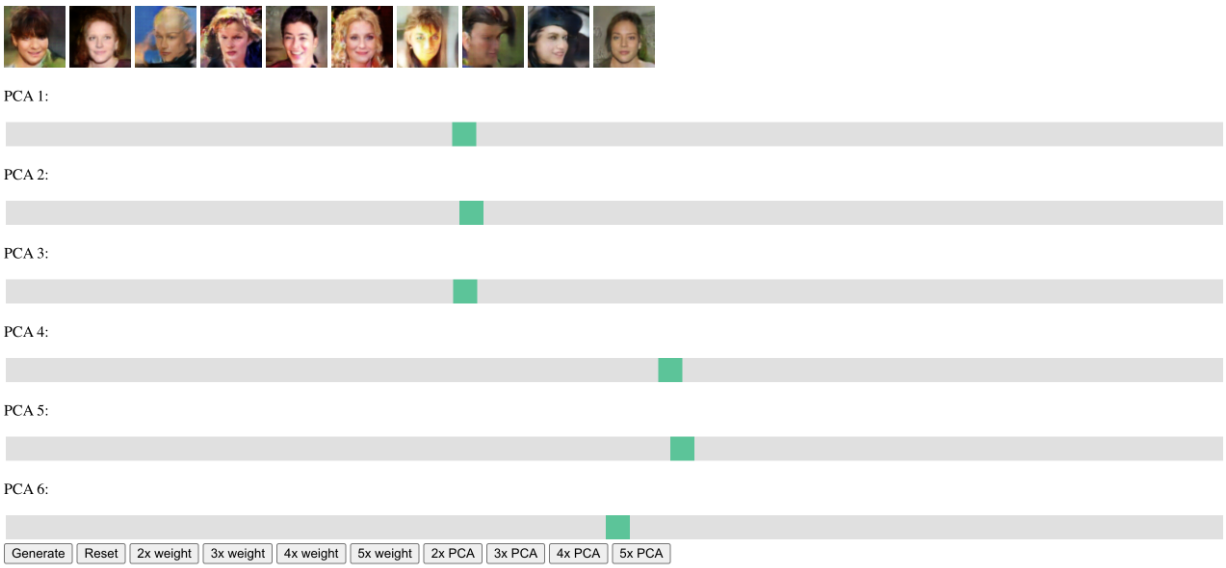

Figure 4.16: Wep Application for exploring the conditional GAN embedding space.

Using the information we learned from the PCA direction we created a web application to be able to manipulate the PCA directions. An image of the web application is shown in Figure 4.16. The web application provides the following controls:

- Slider control of PCA directions.
- Multiplying existing weights.
- Multiple of the PCA weights.

An example of increasing PCA direction 2 which is responsible for generating glasses can be found in Figure 4.17. The web application is hosted on AWS and can be publicly accessed at http://15.222.241.154:5000/.

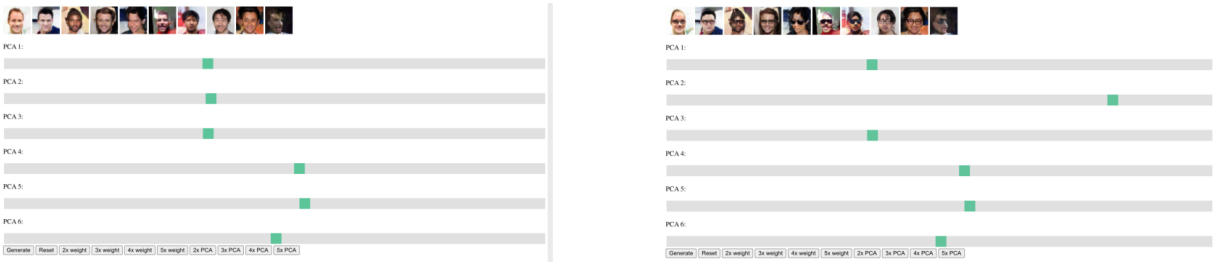

Figure 4.17: Before and after generated images from the web application by changing PCA 2.

4.4 Discriminator Dissection

4.4.1 Network Dissection

Using the *NetworkDissection* technique, we dissect each layer of a PGGAN that has been trained on the LSUN church dataset. The images are generated at 256 x 256 pixels which gives us six different scale layers that is composed of two CNN layers. A breakdown of the different scale layers and their IoU is shown in Figure 4.18. From the breakdown, we can see that the earlier scale layers are focused object outlines while the later scale layers are corresponded with finer details such as grass and colours. The breakdown also makes sense since the way PGGAN is trained is through progressive growing where finer details are added in later layers.

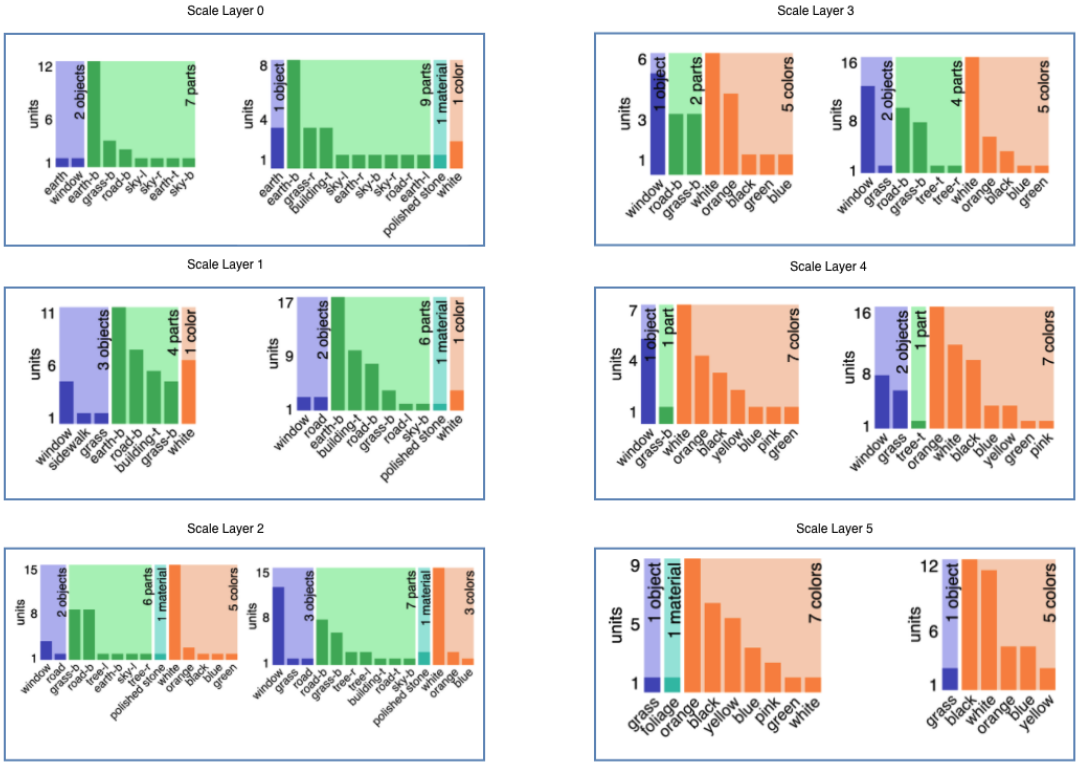

Figure 4.18: Corresponding object semantics of each scale layer.

4.4.2 Object Realness

The object realness of the chosen objects for the LSUN church dataset is shown in Table 3.2. The object realness is calculated using the entire LSUN church dataset which is composed of 126,227 training examples and the same corresponding number of generated images. A high realness indicates a higher increase in the Discriminative score output.

Table 4.2: Object realness output using the LSUN church dataset.

Object	**Realness - (r)**	**Realness - (g)**
Building	0.24	0.22
Tree	0.076	0.19
Road	0.021	0.033
Grass	0.029	0.087
Person	0.025	0
Earth	0.026	0.0088
Sidewalk	0.016	0
Mountain	0.0462	0.045

4.4.3 Custom Dataset

From MTurks a total of 8475 images was found to contain human figure(s). Using the custom dataset we trained both a DCGAN and PGGAN. The DCGAN is set to generate images at 64 x 64 while PGGAN is set to generate at 256 x 256 resolutions. The DCGAN is trained with added Gaussian Noise which has a similar effect as R1 regularization [45]. Examples of identified images are shown in Figure 4.19. The identified images are a mixture where human figures that are the dominate objects or cases where they are more of a background object. During the MTurks experiment, we did not force the workers to be Masters. Masters of MTurks are workers who have demonstrated high excellence across a multitude of tasks. Furthermore, we did not specify any additional qualifications of workers such as age, demographics. As we elected for a no barrier entry of the workers the generated samples should be relatively unbiased. Additionally, since we elected for a unrestricted approach for workers, we only have the label information. From examples of the selected images, we can see from Figure 4.19 selected images contain varying scale of human scales. Future research could be done to force users to select images with more prominent human figures. The result of the DCGAN generation with normal LSUN church is shown in Figure 4.20. The images generated using the custom dataset is shown in Figure 4.21. While we did train a PGGAN on the custom dataset, the training was unstable, and the Generator was unable to generate finer details at higher resolution resulting in training failure.

Figure 4.19: Identified images from MTurks.

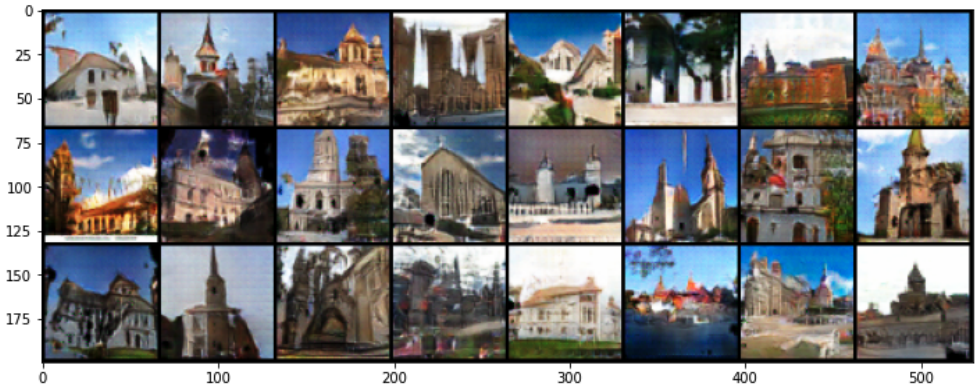

Figure 4.20: Images generated by DCGAN trained on normal church dataset.

4.4.4 Discussion

From the breakdown of the individual scale layers of the Discriminator there were no activations that correspond with a person. The objects that the Discriminator are focusing are buildings, window, earth, grass. Since the role of the Generator is to create images that

Figure 4.21: Images generated by DCGAN trained on custom dataset. Yellow circles indicate the presence of a human figure(s).

fools the Discriminator, if the presence of a person makes no difference to the Discriminator, then there is no incentive for the Generator to create human figures. This is also confirmed via the object realness results. From Table 4.2 we can see that if a person is included in an image the realness does not change.

The object distribution of the generated samples should mimic the object distribution in the real samples. In Table 3.2 we can see that the object distributions is similar but the Generator tend to focus on easier to generate objects. For instance, trees and mountains have a higher presence in the generated sample while roads and person are lower. The experiment with the custom dataset in which every image contains a figure of a person should force the Generator to generate noticeable human figures. The 100% presence of a human figure in the images forced the Generator to try and generate people. However, the training was unstable, and the resulting generated images are of low quality. While the generated images are unstable we can see the emergence of human figures in the images as shown in Figure 4.21. The resulting images might suggest that the reason why objects are amissed is due to the GAN architecture. Both the DCGAN and PGGAN suffered in quality when trained and were unable to converge.

4.5 Data Augmentation

In this section we used the following four datasets: MNIST, Fashion MNIST, Street View House Numbers (SVHN) and CIFAR-10. All four were used in the normalization experiment but for the data augmentation we used FashionMNIST and SVHN as MNIST was too simple and CIFAR-10 is a complicated dataset for GANs.

4.5.1 Normalization

For the test of normalization we used all four datasets. We used two different conditional DCGAN architectures as MNIST and Fashion-MNIST are grayscale images, and the image size is different from SVHN and CIFAR-10. When testing the effect of custom normalization both the generated and real image must be unnormalized first and then normalized according to the custom mean. The mean and standard deviation of the different datasets are captured in Table 4.3.

Table 4.3: Mean and Standard Deviation of the different datasets.

Dataset	**Mean**	**Standard Deviation**
MNIST	0.1307	0.3081
Fashion-MNIST	0.286	0.353
CIFAR-10	(0.4914, 0.4822,0.4465)	(0.247, 0.2435, 0.2616)
SVHN	(0.4377, 0.4438, 0.4728)	(0.198, 0.201, 0.197)

4.5.2 Augmentation Comparison

For image augmentations we train a GAN model for each combination of the four augmentations: rotate, translation, scale, and horizontal flip. We also trained a GAN model without any image augmentations for comparisons later. Since GANs are notoriously difficult and we are applying image augmentations during the process we elected to add Gaussian noise into the Discriminator module. By adding Gaussian noise, we get a similar effect as R1 regularization which has been proven to stabilize training and improve the result [45]. Instead of using all four datasets, we used Fashion-MNIST and SVHN. MNIST was not considered as our chosen classifier (ResNet50) can easily achieve high accuracy and

CIFAR-10 is a difficult dataset to generate realistic images for without leveraging more complex GAN architectures. For the parameters used for the augmentations refer to Table 4.4. The parameters are chosen to prevent violation of the label due to transformation.

In our experiment we focus on breaking down the comparison of augmentation as baseline vs. (real + generated) + image augmentation vs. real + generated augmented images where Generated images are 50% of the real training set. The baseline is established using a ResNet50 that is trained on the real dataset. The difference between the second and third method is that in the second method we apply image augmentation on real and generated dataset while the third is real with images that are already augmented.

Table 4.4: Augmentation parameters. Translation uses a range to indicate the amount of shift with respect to the size of the image. Horizontal flip indicates the chance of flipping an image.

Augmentation	**Parameter**
rotate	10 degrees
translation	(0.1, 0.1)
scale	(0.9, 1.1)
Horizontal Flip	0.5

4.5.3 Results

Normalization

The result of the normalization comparison is captured in Table 4.5.

Augmentation Comparison

The results of the data augmentation comparison is captured in Table 4.6 and 4.7 where R is for rotation, T is for translation, S for scale and H for horizontal flip. From the two tables we can see that for FashionMNIST the resulting accuracy of the classifier was higher for the "Normal + GAN transformatio" scenario. It was able to beat out both "Normal" and "(Normal + GAN) + transformation" on a consistent bases. However, for the SVHN dataset the same transformation ended up performing worse. This is a result

Table 4.5: Resulting FID score of the different datasets.

Dataset	**FID**
MNIST	12.58971
MNIST with custom	11.83819
Fashion-MNIST	40.8258
Fashion-MNIST with custom	40.0888
CIFAR-10	60.06015
CIFAR-10 with custom	61.6957
SVHN	163.6257
SVHN with custom	191.90438

Table 4.6: Data Augmentation result MNIST.

Transformation	**Normal**	**(+ GAN) + transformation**	**+ GAN transformation**
Baseline	90.32	90	**90.36**
R	90.32	89.35	**90.4**
T	88.77	89.35	90.49
S	90.1	89.74	**90.49**
H	**90.38**	90.25	**90.38**
R + T	87.88	87.95	**90.47**
R + S	88.77	89.09	**90.64**
R + H	89.57	89.58	**90.12**
T + S	88.14	88.23	**90.33**
T + H	89.24	89.07	**90.34**
S + H	89.95	89.69	**90.58**
R + T + S	87.31	86.85	**90.55**
R + T + H	87.7	88.04	**90.35**
R + S + H	88.85	88.86	**90.6**
T + S + H	87.52	87.84	**90.42**
R + T + S + H	86.95	87.09	**90.72**

Table 4.7: Data augmentation result SVHN.

Transformation	**Normal**	**(+ GAN) + transformation**	**+ GAN transformation**
Baseline	**91.2646**	90.7114	90.3427
R	91.134	**91.2262**	91.1186
T	93.4696	**93.362**	91.0149
S	**91.3722**	90.9919	91.2339
H	89.8663	89.5129	**90.4425**
R + T	**92.7704**	92.3479	90.7883
R + S	**91.1494**	91.0418	91.0034
R + H	90.1429	90.2044	**90.7575**
T + S	**93.0317**	93.0278	90.7691
T + H	**91.8946**	91.7409	91.2262
S + H	90.2581	90.0277	**90.8305**
R + T + S	**91.9829**	91.8523	90.6615
R + T + H	90.4656	90.6653	**90.723**
R + S + H	89.9892	89.9623	**91.1378**
T + S + H	91.5373	**91.94**	91.1993
R + T + S + H	90.3081	90.1967	**90.8574**

of the Generator not being good enough to generate images that are comparable to the real images. Since the generated images are not good enough, we can see that by training with them it can result in degradation of the classification accuracy. We can can confirm this by looking at Table 4.5, the SVHN dataset had a FID score of 191 as compared to the FashionMNIST's 40.82.

4.6 Time Series Generation

4.6.1 Model

Since GANs can be difficult to train we leveraged the architecture from RGAN [14] as the base model. The RGAN uses long short-term memory (LSTM) [43] in both the Discriminator and Generator module to capture time dependent information. A given LSTM layer will compute the following for each element in the input sequence:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_i i + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_i f + W_{hf}h_{t-1} + b_{hf} \\ g_t &= tanh(W_{ig}X_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{4.1}$$

where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, h_{t-1} is the hidden state of the layer at time $t-1$ and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates. σ is the sigmoid function and $\odot$ is the Hadamard product. For more details regarding LSTM, please refer to [43].

4.6.2 Generator

The architecture of the Generator is shown in Figure 4.22. The latent dimension chosen is 100, the number of features in the hidden state h_t is 32. A fully connected layer is used to convert the latent input at each time step into the desired data feature dimension.

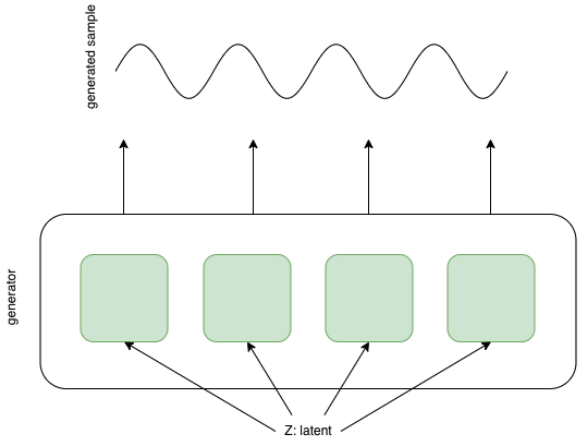

Figure 4.22: Generator architecture.

4.6.3 Discriminator

The architecture of the Discriminator is shown in Figure 4.23. The parameters of the Discriminator are designed as a reflection of the Generator and thus have the same number of features (32) at each hidden state h_t. A fully connected layer is also used to determine if the data at each time step is generated or fake.

4.6.4 Training

The datasets are standardized using a min-max scaler prior to splitting. The datasets were then split into training and testing sets using 80% training and 20% testing. The testing set is used for evaluation purposes that we will cover in the next section. The different GAN models were then trained for 1000 epochs and a standard Adam optimizer [25] with the default configuration of 0.0002 for the learning rate and 0.5, 0.999 for beta1 and beta2 was used. With respect to the number of LSTM layers we varied the number of layers from 1 to 3 and captured the best model according to our evaluation metrics.

An experiment that was not carried out in related works is the evaluation of performance with respect to the sequence length. As the sequence length increases, it becomes more difficult for GANs to generate more realistic samples. This could be akin to generating

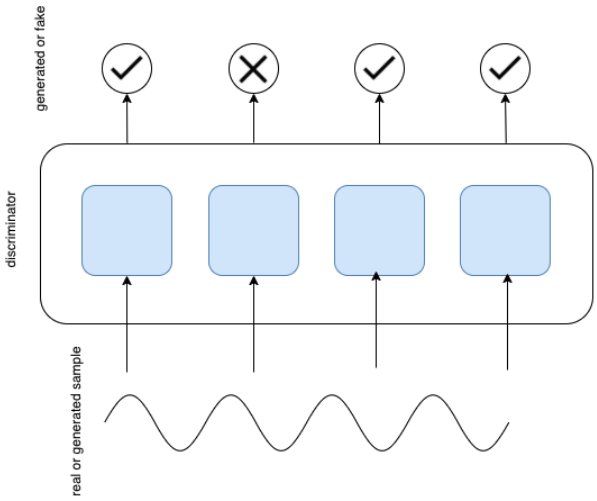

Figure 4.23: Discriminator architecture.

higher resolution images. To verify the performance of our proposed approaches, for each of the dataset we tested the performance starting from a sequence length of 4 and increasing the length by 4 each time until we reached 24. Alternatively, we could have chosen a starting length of 2 and incremented by 2 each time but we felt that sequence length of 2 to contain less information than a sequence length of 4.

To ensure that the *Discriminative Score* was not coincidental we trained the same models from scratch 5 times and calculated its score for each iteration. We then averaged the averages across the multiple iterations to obtain the final *Discriminative Score*.

4.6.5 Evaluation

For GANs that are trained for image generation, standard evaluation methods such as Inception Score (IS) and Frechet Inception Distance score (FID) exit but no standard evaluation metrics are available for time series. To evaluate the performance of time series generated by GANs we need to consider the *fidelity* and *diversity* of the generated sample.

Fidelity. Fidelity measures how distinguishable the generated samples are from the real.

Diversity. The samples should be diverse and 'cover' the original sample.

To evaluate *fidelity* and *diversity* we apply the evaluation metric of PCA, t-SNE [54] and Discriminative score introduced in [60].

PCA - t-SNE. PCA and t-SNE can be applied on both the original and generated samples by flattening the temporal dimension. The result of applying the two techniques can be visualized in a 2-dimensional space allowing us to visually evaluate the *diversity* of the generated samples. To create the PCA and t-SNE visualization 1000 samples are generated, and 1000 real samples are used.

Discriminative Score. To determine the *fidelity* of the generated samples we use a time series classification network composed of 2 LSTM layers trained to distinguishing between real and generated samples. The classification network is trained on the same training set that was used to train the GAN models and a training set of equal size of generated data. This is a difference between the *Discriminative Score* used in our study and [60]. In [60] GANs are trained using all the data with no testing set but a testing set was created for the post-hoc classifier. Rather than training the GANs with all the data we chose to use the training set only and reserving the testing for more accurate scoring since the testing data should not be leaked to the GANs. We also withheld a separate generated testing set that is used in conjunction with the real testing set to evaluate the performance of the classification. The lower the scoring of the classification model on the validation set, the higher the *fidelity* of the generated samples since the network was not able to easily discern between generated and real samples.

4.6.6 Results

In this section, we present the results of the *Discriminative score* of the tested models and visualization of their generated samples.

Visualization

The result of PCA and t-SNE on the generated and real samples are shown in Figure 4.24. Figure 4.24 captures the result of applying a 2-dimensional reduction on sequence length of 4 data from both the *energy* and *stock* data.

Discriminative Score

The *Discriminative score* of the three different models: base model, *sDTW-p*, *sDTW-m* across all sequence length from 4 to 24 are shown in Table 4.8 and Table 4.9.

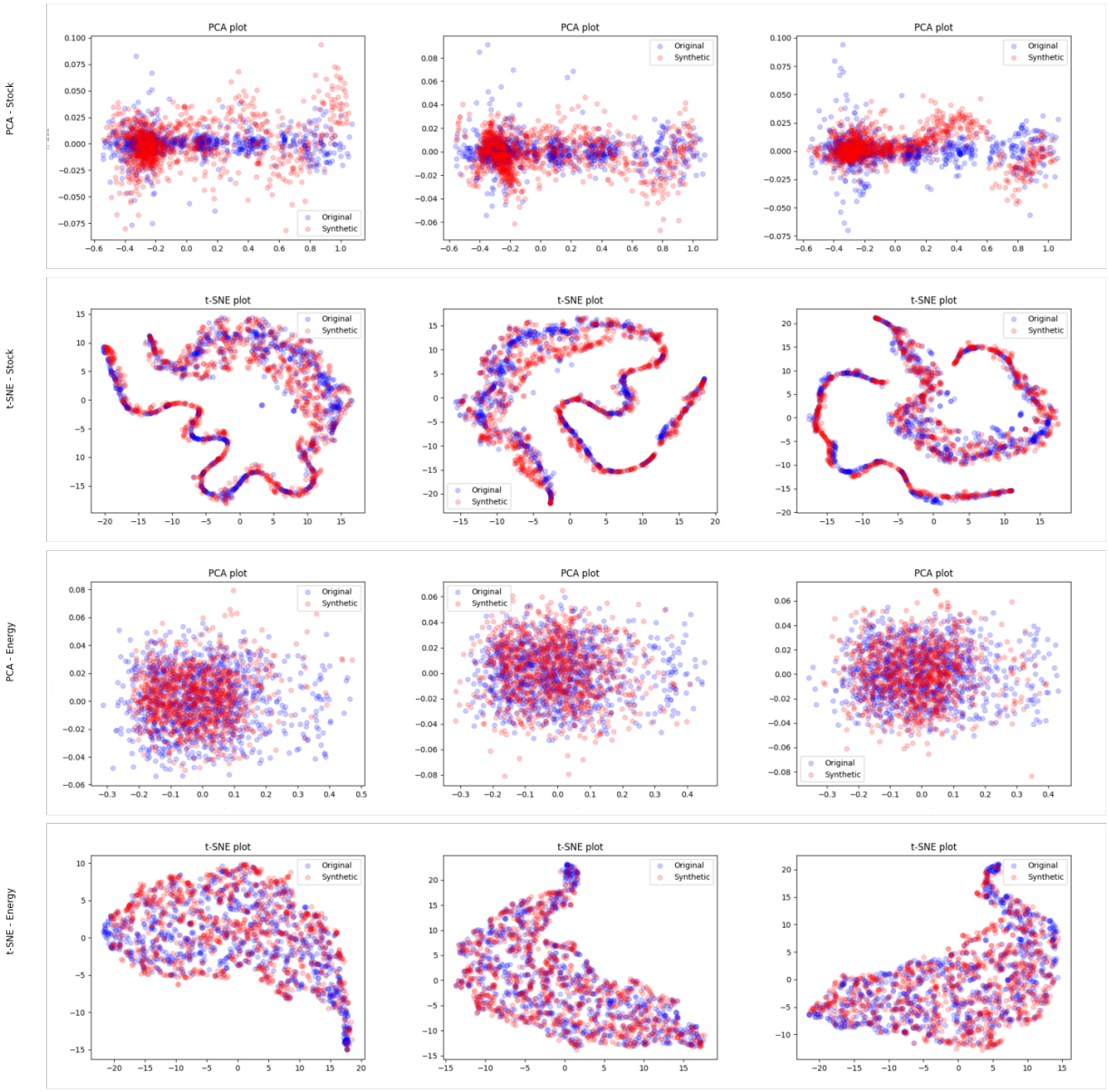

Figure 4.24: PCA and t-SNE plots of both the generated and real samples. The shown figures are taken from models trained using a sequence length of 4. Starting from right to left are the base model, *sDTW-p* and *sDTW-m*.

Table 4.8: Discriminative Score (lower the better) tested on the *stock* dataset.

Model	Dataset	4S	8S	12S	16S	20S	24S
Base	Stock	0.877	0.852	0.94	0.962	0.945	0.909
Base + sDTW-p	Stock	0.852	0.852	**0.862**	**0.924**	**0.863**	0.875
Base + sDTW-m	Stock	**0.819**	**0.812**	0.868	0.969	0.885	**0.862**

Table 4.9: Discriminative Score (lower the better) tested on the *energy* dataset.

Model	Dataset	4S	8S	12S	16S	20S	24S
Base	Energy	0.959	0.986	**0.995**	0.998	0.998	0.999
Base + sDTW-p	Energy	0.960	**0.985**	0.996	**0.997**	**0.996**	**0.997**
Base + sDTW-m	Energy	**0.949**	0.991	0.996	0.998	0.999	0.998

4.6.7 Discussion

Discriminative Score

From Table 4.8 and 4.9 we can see that the two models *sDTW-p* and *sDTW-m* consistently outperformed the base model with the exception of the *energy* data with a sequence length of 12. However, the difference was very negligible as it was 0.9952 vs. 0.9957 from *sDTW-p*. The biggest difference in *Discriminative score* was the *stock* data with a sequence length of 12. The base model had a score of 0.9404 and while *sDTW-p* was 0.8627, this is equivalent to a 9% improvement. While the *stock* data saw major improvements across different sequence lengths the *energy* dataset had less improvements. The biggest difference was 0.9992 vs. 0.99753 which is a 0.245% improvement. The lower improvement for *energy* data could be due to the difference in data features. *Stock* had a feature dimension of 6 while *energy* had a feature dimension of 28. The larger feature dimension makes it difficult for the Generator module to generate realistic samples.

Visualization Result

Intuitively, a lower *Discriminative score* should be correlated with samples that are close to each other in the 2-dimensional space after PCA or t-SNE is performed. We can see

from Figure 4.18 that in the PCA plot of *sDTW-m* (last column from the left) for the stock dataset the synthetic sample and the real sample are much closer. Similarly, in the t-SNE plot, the samples generated by *sDTW-m* hugged the real samples much closer. For the energy data since the difference in *Discriminative score* was minimum this meant the PCA and t-SNE plot of the three models would be extremely similar. Additionally, the visualization via PCA/t-SNE confirms our discriminative score. A lower discriminative score would mean the synthetic samples should be closer and harder to distinguish from the real samples as it is shown in our visualization.

***sDTW-p* vs. sDTW-m**

In the previous sections, we showed that the introduction of *sDTW* into the loss function led to a consistent improvement across different time sequences, but the two loss functions are better at different sequence lengths. However, the performance of the two different loss functions is quite even with each performing better for different sequences lengths. However, as the sequence length increases the performance of *sDTW-m* tend to be worst. This is especially true for more complicated dataset, in Table 4.9 beside sequence length of 4 the performance of sDTW-p was better for the remaining sequence lengths. This suggests that for lower sequence length *sDTW-m* can potentially help to increase the diversity of the generated sample while still following the original sample distribution but as the sequence length increases this might cause the Generator to generate samples that are too far from the real sample and thus the *Discriminative score* suffers.

Chapter 5

Conclusion

The primary objective of this research was to increase understandability and interpretability of the GAN black box model through various methods such as popular off the shelf explainability frameworks and more powerful but custom methods such as network dissection and our own novel methods such as Object Realness, Discriminator Dissection, Forced Object Generation and conditional embedding space exploration. The secondary objective was to provide a comprehensive comparison of GANs with data augmentation. The last objective was to improve GAN generation for time series.

In the first objective, through a comparison of various off the shelf explainability frameworks we found that SHAP was able to consistently provide comprehensive insights into the decisions of the Discriminator module. While the frameworks provided an excellent starting point for understanding the Discriminator. Our own methods of Object Realness and forced data generation provided more details into the existence of mode collapse. Our exploration of the embedding space of the conditional variant revealed human interpretable semantics that can be manipulated through the PCA direction. To leverage this finding we created a web application that allow users to freely manipulate the generated image through the PCA direction.

Our second objective revealed interesting aspects of techniques in traditional CNN that could be applied to GAN. For grayscale images we found that normalizing according to custom mean and STD can lead to better FID score while grayscale images that were generated through data augmentation can consistently lead to better classification accuracy. However, the coloured images did not experience the same finding.

In our last objective, we introduced two new loss functions *sDTW-p* and *sDTW-m* based on *sDTW* that can be used to improve the performance of time series generation.

We thoroughly tested the performance of the loss function across different time sequence lengths and evaluated the results based on *fidelity* and *diversity* to prove that the new loss functions consistently outperform against the baseline. We also showed that the two loss functions can have different use-cases based on the length of the sequences.

The potential of the research finding as well as work that can be done in the future can be summarized as follows:

- Examining the effect of inverse latent for conditional GANs.
- Expand the object realness and force generation to GANs that generate videos.
- Apply XAI frameworks to video generation GANs.
- Examine whether other existing time series GANs can benefit from the introduction of our novel loss functions.
- Compare the differences between leveraging LSTM vs. transformers for time series generation as transformers can potentially lower the computational time.
- A shaped-based DTW approach can be examined to compare the effect with the sDTW loss function.

All the code used in this experiment as well procedure for reproducing the results can be found in https://github.com/NullCodex/thesis.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3319–3327, 2017.

[3] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020.

[4] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[5] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Bolei Zhou, Hendrik Strobelt, and Antonio Torralba. Seeing what a gan cannot generate. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.

[6] Eoin Brophy, Maarten De Vos, Geraldine Boylan, and Tomás Ward. Multivariate generative adversarial networks and their loss functions for synthesis of multichannel ecgs. *IEEE Access*, 9:158936–158945, 2021.

[7] Fred B. Bryant and Paul R. Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis. 1995.

[8] Luis M. Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97, 2017.

[9] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for Anomaly Detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[10] Marco Cuturi and Mathieu Blondel. Soft-dtw: A differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 894–903. JMLR.org, 2017.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[13] Khaled El Emam, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin. A systematic review of re-identification attacks on health data. *PLOS ONE*, 6(12):1–12, 12 2011.

[14] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *ArXiv*, abs/1706.02633, 2017.

[15] Leon Gatys, Alexander Ecker, and Matthias Bethge. A neural algorithm of artistic style. *Journal of Vision*, 16(12):326–326, Sep 2016.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5769–5779, Red Hook, NY, USA, 2017. Curran Associates Inc.

[18] David Gunning and David Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, Jun. 2019.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc.

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[23] Young-Seon Jeong, Myong K. Jeong, and Olufemi A. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9):2231–2240, 2011. Computer Analysis of Images and Patterns.

[24] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[26] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[28] Lan Lan, Lei You, Zeyang Zhang, Zhiwei Fan, Weiling Zhao, Nianyin Zeng, Yidong Chen, and Xiaobo Zhou. Generative adversarial networks and its applications in biomedical informatics. *Frontiers in Public Health*, 8:164, 2020.

[29] Heyi Li, Yuewei Lin, Klaus Mueller, and Wei Xu. Interpreting galaxy deblender gan from the discriminator's perspective. In George Bebis, Zhaozheng Yin, Edward Kim, Jan Bender, Kartic Subr, Bum Chul Kwon, Jian Zhao, Denis Kalkofen, and George Baciu, editors, *Advances in Visual Computing*, pages 239–250, Cham, 2020. Springer International Publishing.

[30] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[31] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[32] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[33] Mehran Maghoumi, Eugene Matthew Taranta, and Joseph LaViola. *DeepNAG: Deep Non-Adversarial Gesture Generation*, page 213–223. Association for Computing Machinery, New York, NY, USA, 2021.

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[35] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. 2014.

[36] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training, 2016.

[37] Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

[38] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[39] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[40] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. *KDD*, 2012:262–270, Aug 2012.

[41] Manassés Ribeiro, André Eugênio Lazzaretti, and Heitor Silvério Lopes. A study of deep convolutional auto-encoders for Anomaly Detection in videos. *Pattern Recognition Letters*, 105:13–22, 2018.

[42] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[43] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

[44] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[45] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2234–2242, Red Hook, NY, USA, 2016. Curran Associates Inc.

[46] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[47] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019.

[48] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3145–3153. JMLR.org, 2017.

[49] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

[50] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[52] Luke Taylor and Geoff S. Nitschke. Improving deep learning with generic data augmentation. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547, 2018.

[53] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

[54] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[55] Abdul Waheed, Muskan Goyal, Deepak Gupta, Ashish Khanna, Fadi Al-Turjman, and Placido Rogerio Pinheiro. Covidgan: Data augmentation using auxiliary classifier gan for improved covid-19 detection. *IEEE access : practical innovations, open solutions*, 8:91916–91923, May 2020.

[56] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[57] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[58] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[59] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, 58:101552, Dec 2019.

[60] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[61] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[62] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for GAN training. *arXiv preprint arXiv:2006.02595*, 2020.

[63] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.