Benevolent and Malevolent Adversaries: A Study of GANs and Face Verification Systems

by

Ehsan Nazari

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Computer Science
in
Electrical and Computer Engineering

Examining Committee

The following served on the Examining Committee for this thesis.

External Member: Nathalie Japkowicz
Professor and Department Chair, Computer Science
Department of Computer Science
American University

Internal Member(s): Paria Shirani
Assistant Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Supervisor(s): Paula Branco
Assistant Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Guy-Vincent Jourdan
Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Cybersecurity is rapidly evolving, necessitating inventive solutions for emerging challenges. Deep Learning (DL), having demonstrated remarkable capabilities across various domains, has found a significant role within Cybersecurity. This thesis focuses on benevolent and malevolent adversaries. For the benevolent adversaries, we analyze specific applications of DL in Cybersecurity contributing to the enhancement of DL for downstream tasks. Regarding the malevolent adversaries, we explore the question of how resistant to (Cyber) attacks is DL and show vulnerabilities of specific DL-based systems.

We begin by focusing on the benevolent adversaries by studying the use of a generative model called Generative Adversarial Network (GAN) to improve the abilities of DL. In particular, we look at the use of Conditional Generative Adversarial Network (CGAN) to generate synthetic data and address issues with imbalanced datasets in cybersecurity applications. Imbalanced classes can be a significant issue in this field and can lead to serious problems. We find that CGANs can effectively address this issue, especially in more difficult scenarios. Then, we turn our attention to using CGAN with tabular cybersecurity problems. However, visually assessing the results of a CGAN is not possible when we are dealing with tabular cybersecurity data. To address this issue, we introduce AutoGAN, a method that can train a GAN on both image-based and tabular data, reducing the need for human inspection during GAN training. This opens up new opportunities for using GANs with tabular datasets, including those in cybersecurity that are not image-based. Our experiments show that AutoGAN can achieve comparable or even better results than other methods.

Finally, we shift our focus to the malevolent adversaries by looking at the robustness of DL models in the context of automatic face recognition. We know from previous research that DL models can be tricked into making incorrect classifications by adding small, almost unnoticeable changes to an image. These deceptive manipulations are known as adversarial attacks. We aim to expose new vulnerabilities in DL-based Face Verification (FV) systems. We introduce a novel attack method on FV systems, called the DodgePersonation Attack, and a system for categorizing these attacks based on their specific targets. We also propose a new algorithm that significantly improves upon a previous method for making such attacks, increasing the success rate by more than 13%.

Acknowledgements

My journey pursuing a master's at the University of Ottawa has been a profound experience. The challenges I encountered in my research could have been insurmountable had it not been for the unwavering guidance, support, and leadership of my supervisors, Professor Branco and Professor Jourdan. Their mentorship transformed mere ideas into meaningful contributions to the research community.

First and foremost, I would like to express my heartfelt appreciation to Professor Branco. Her consistent support, demonstrated through her kindness, meticulous attention to detail, and the precision with which she monitored my progress during our weekly meetings, was indispensable.

Furthermore, I owe a deep debt of gratitude to Professor Jourdan. His comprehensive guidance throughout my master's has been invaluable, and the insights and life lessons he shared extended beyond the academic realm, enriching my personal journey.

I would like to express my gratitude to Bill Aiken, a member of our lab, for his valuable comments and insights on our final work (Chapter 5). I also extend my thanks to José Charlos Mondragon for his assistance with the plots in Chapter 5.

I also acknowledge the silent mentors whose work in Machine Learning (ML) and mathematics has profoundly shaped my understanding. The authors of the reference books I studied, and creators like Grant Sanderson (3Blue1Brown), Josh Starmer (StatQuest), and Jason Brownlee (machinelearningmastery), have indirectly contributed to my learning through their online platforms, for which I am grateful.

Lastly, a special mention to my parents. Their presence in my life is beyond academic pursuits. I owe my very being to them. Their unconditional love, coupled with their unwavering belief in my decisions and endeavors, has been the bedrock of my journey. I cannot thank them enough for everything they've given and continue to give.

Table of Contents

List of Tables

List of Figures

Abbreviations

CAS-real Classification Accuracy Score on real data xiv, 44, 51, 52, 56, 57, 59, 60, 67, 76

CAS-syn Classification Accuracy Score on synthetic data xiv, 44, 52, 56, 57, 59, 60, 67, 76

CDS Confidence and Diversity Score xiv, 45, 54, 55, 57, 59, 60, 67, 69, 73, 76

CGAN Conditional Generative Adversarial Network iv, vii, xi, xiii, 2, 3, 9, 13, 14, 16–20, 24–32, 34–37, 43, 44, 51, 52, 56, 66, 67, 118

DL Deep Learning iv, 1–3, 8, 118

FCD Fréchet Confidence and Diversity xiv, 40, 46, 55–57, 59, 60, 67, 69, 72, 73, 76

FID Fréchet Inception Distance xiv, 40, 45, 46, 53–55, 57, 60, 72, 73, 76

FM Face Mapper xv, 83, 86, 88, 96, 97, 99, 110–112, 116

FV Face Verification iv, xv, 3, 5, 10, 11, 80–88, 90–92, 96, 98, 105, 106, 109, 114, 116, 118, 119

GA Genetic Algorithm xii, xvi, 92, 93, 95, 107, 108, 111

GAN Generative Adversarial Network iv, vii, xiii, xiv, 2, 3, 5–9, 12–16, 37–50, 53–59, 64–67, 69, 74, 76, 78, 79, 84, 103, 118, 119

IS Inception Score xiv, 40, 45–47, 53, 54, 56, 57, 60, 73, 76

ML Machine Learning v, 1, 2, 5–8, 11, 12, 39

WGAN Wasserstein GAN 40

Chapter 1

Introduction

Within cybersecurity, DL has risen as an instrumental solution to address the ever-evolving cyber threats. While DL algorithms have demonstrated their effectiveness in detecting intricate patterns from vast datasets, they also present unique vulnerabilities. This thesis explores the domain of DL, emphasizing its applications, potentials, and challenges, primarily within the realm of cybersecurity.

1.1 Deep Learning in Cybersecurity

Security holds importance in both physical and digital aspects of our lives. We implement different protective measures, such as knife sheaths, car seat belts, ship lifeboats, and garden fences to ensure safety. Similarly, in the digital realm, security measures are of significance. Passwords serve as a shield for personal information, antivirus software defends against threats, firewalls prevent unauthorized access, and VPNs ensure secure browsing. This field, focusing on digital security, is known as Cybersecurity.

Cybersecurity has unique characteristics and requires distinct approaches compared to the physical world. For instance, the rise of credit cards has led to the emergence of credit card fraudulent transactions, which pose new challenges in Cybersecurity. Detecting such transactions has become a crucial task, and various methods have been proposed to address it. ML, particularly DL, is increasingly being used to tackle issues related to fraudulent transactions.

ML, a subset of Artificial Intelligence, empowers computer programs to extract rules and patterns from provided data without direct human intervention, implicitly capturing

the underlying patterns. Within ML, DL algorithms are designed to automatically learn intricate patterns and features from large datasets. DL has witnessed outstanding growth in recent years, demonstrating enhanced performance and efficiency and surpassing human-level accuracy in several tasks [1, 41, 51, 79, 151].

A basic deep neural network consists of layers of artificial neurons stacked on top of each other. Once trained, each layer within the network offers a new representation of given data. In essence, DL serves as a methodology for representation learning, introducing complex concepts expressed in terms of simpler representations. In other words, each layer of a deep neural network introduces a different representation of the given data. This approach empowers computers to construct intricate concepts by leveraging simpler ones [36]. The extracted representations obtained from DL models are highly valuable in various applications. For instance, in computer vision, these representations enable accurate image classification; in natural language processing, they can be utilized for sentiment analysis; in generative models like GANs, the generated representations resemble the distribution of the original data [4]. These representations have opened up possibilities for numerous downstream tasks, making DL an incredibly valuable asset. The recent advancements in DL led to its integration into numerous digital tools we utilize today, such as laptops, smartphones, game consoles, traffic cameras, border security cameras, self-driving cars, and many other applications. DL has become an omnipresent technology in our surroundings.

This thesis examines the utilization of DL, primarily within a Cybersecurity context. It is dedicated to unraveling different facets of DL. It begins by focusing on understanding and improving DL techniques. Then, our investigation goes beyond the enhancement of DL techniques. We attach equal importance to discovering and illustrating its vulnerabilities. We aim to highlight the potential weak spots of DL when applied to Cybersecurity. In essence, this thesis is a compilation that explores the advancements as well as the shortcomings of DL, mainly as they relate to the field of Cybersecurity.

1.2 Overview of Thesis

In this thesis, we address the problem of imbalanced datasets in Cybersecurity applications by utilizing GAN-based upsampling, specifically focusing on the suitability of CGAN as an upsampling strategy. Our research showcases consistent improvements in performance for minority classes, contributing to a better understanding of how CGAN can effectively address class imbalance issues. Additionally, we introduce AutoGAN to efficiently train GANs on both imagery and tabular data, minimizing human inspection during training and achieving comparable or superior results in tabular Cybersecurity datasets. Lastly, we

expose vulnerabilities in DL-based FV systems, presenting the DodgePersonation Attack as a means to deceive FV systems.

DL can be considered a versatile tool for executing specific tasks, but its potential extends beyond that. In Chapter 3 of the thesis, we explore how DL can be used to enhance itself. Through a series of comprehensive experiments, we investigate the upsampling capabilities of a generative model called a GAN in a downstream classification task. Our main emphasis lies in utilizing CGAN for data synthesis, specifically aiming to tackle the challenge of imbalanced datasets, particularly in the field of Cybersecurity applications. The problem of imbalanced classes holds significant importance in this domain and can lead to severe consequences [57]. The results of our study highlight the consistent improvement in performance for minority classes through CGAN-based upsampling, particularly in challenging scenarios. Our research contributes to a better understanding of how CGAN can effectively address class imbalance issues. This chapter exemplifies the utilization of DL to effectively tackle Cybersecurity problems suffering from the class imbalance problem.

Chapter 3 demonstrates that, when working with tabular Cybersecurity datasets, the visual assessment of the CGAN generator's outputs is not possible. This challenge serves as the foundation for Chapter 4, where the methodology called AutoGAN is introduced to efficiently train a GAN on both imagery and tabular data, addressing this issue. In this chapter, the focus is on minimizing human inspection during GAN training. AutoGAN opens new possibilities for utilizing GANs in tabular datasets, including Cybersecurity datasets that are not of an imagery nature. Extensive experiments involving various tabular and imagery datasets demonstrate that AutoGAN achieves comparable or superior results compared to other alternative methods.

In Chapter 5 we shift our focus to examining the security and robustness of DL models in the context of face recognition. In an early work by Goodfellow et al. [39], it was demonstrated that DL models can be easily fooled by adding imperceptible perturbations to an image, leading to misclassifications. This pioneering research opened the door to adversarial attacks, highlighting the vulnerability of DL models. Chapter 5 aims to expose new vulnerabilities in DL-based facial recognition systems. DodgePersonation Attack, a comprehensive and versatile approach to understanding attacks on FV systems, is introduced, alongside a taxonomy for categorizing adversarial attacks based on their specific targets. A novel algorithm is proposed that surpasses the Master Face Attack of a previous study, increasing the coverage by more than 13%.

1.3 Publications

The work in Chapter 3 is based on the following two papers:

- Ehsan Nazari, Paula Branco, and Guy-Vincent Jourdan. Using cgan to deal with class imbalance and small sample size in cybersecurity problems. In *2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE, 2021

- Ehsan Nazari and Paula Branco. On oversampling via generative adversarial networks under different data difficulty factors. In *Proceedings of the Third International Workshop on Learning with Imbalanced Domains: Theory and Applications*, volume 154 of *Proceedings of Machine Learning Research*, pages 76–89. PMLR, 17 Sep 2021

The work in Chapter 4 is based on the following paper:

- Ehsan Nazari, Paula Branco, and Guy-Vincent Jourdan. Autogan: An automated human-out-of-the-loop approach for training generative adversarial networks. *Mathematics*, 11(4), 2023

Chapter 2

Background

In this chapter, we explore the concepts that form the basis of this thesis. Initially, we investigate the challenges related to the application of appropriate data for machine learning models in classification tasks. There are instances where there might be a lack of relevant data. In some cases, datasets may display imbalances. Furthermore, we present a tool, GAN, which possesses the potential to enhance and optimize our datasets. These concepts will lay the groundwork for Chapter 3 and Chapter 4. Finally, we shift our focus to FV which serves as the foundation for the Chapter 5 of the thesis.

2.1 Data Difficult Factors

The foundation of ML relies on two crucial elements: ML algorithms and the data used by these algorithms. An important question arises: which of these elements carries more significance? According to a paper authored by Michele Banko et al. [5], it was observed that various ML algorithms exhibited nearly identical performance when tackling a complex problem of natural language disambiguation, provided they were given sufficient data. This finding suggests that research communities may need to reassess how they allocate resources between refining algorithmic techniques and enriching the data corpus, as supported by Halevy et al. [47].

Datasets have their set of imperfections that can adversely affect the efficacy of ML algorithms that deploy them. Numerous challenges arise when employing data for ML models. For instance, there might be instances of missing data or features, as pointed out by [33]. Another concern is noisy data, which can manifest as mislabeling, termed class

noise, or as attribute noise when there are corruptions in data attributes, as noted by [131]. Additionally, rare data instances can result in small disjuncts, which are subsets of data more prone to mistakes than their larger counterparts in learning systems, as described by [56]. Below, we will outline four additional factors related to data difficulty that are examined in Chapter 3 and Chapter 4.

Small Sample Size Problem. The volume of data is crucial for ML models. Halevy et al. [47] emphasize the significance of this. Fast forward, the recent developments in the Computer Science domain, notably the emergence of Transformers [137] that paved the way for cutting-edge models like BERT [63] and GPT [104], would not have been possible without substantial data. Conversely, when dealing with a limited sample size, ML models face significant challenges [143].

To tackle the challenge of limited data, several strategies have been recommended. One notable method is transfer learning, where a model previously trained on one dataset is further trained on another. This leverages existing patterns from the original dataset, often proving more effective than starting with a completely untrained model [148]. Additionally, data augmentation improves a model's generalization capacity when faced with a sparse dataset. For image-related data, this can include methods such as cropping, rotating, scaling, adversarial training, and even generating synthetic data using GANs [123].

Class Overlap Problem. In the realm of data space, instances from varying classes can often occupy a shared area, reflecting similarities in their features. This overlap, known as class overlap, is where unclear boundaries or similar attributes emerge between classes, adding complexity to classification tasks [42, 126]. Such a phenomenon becomes even more challenging when class imbalances are present, as noted in [113, 138].

To address the problem of class overlap, methods such as feature engineering and feature selection can be employed. These techniques aid in diminishing irrelevant or redundant features, which could potentially introduce noise and exacerbate the overlap between classes. A notable example of a feature extraction method is Principal Component Analysis (PCA) [120].

Low Data Dimensionality Problem. When we possess more features for a specific phenomenon within a dataset, we are likely to cover a broader range of its aspects. This enhances an ML model's ability to discern patterns. Conversely, with fewer features for each sample, ML models may struggle to comprehensively understand the data for subsequent tasks, such as classification.A clear solution would be to gather more insights from various facets of the concept to address this issue.

Class Imbalance Problem. Class imbalance describes a situation in which a dataset, containing a minimum of two classes of data, displays an imbalance [56]. Specifically, one

class has a significantly greater number of samples compared to the other. This imbalance causes the minority class to be under-represented, making it a challenge for a ML model to extract features effectively and accomplish optimal performance [42, 66, 126]. In the subsequent paragraphs, a set of solutions for this issue is presented. The imbalance between classes poses a challenge in various practical applications. To address this issue, pre-processing methods have emerged as a prevalent approach, aiming to alter the distribution of the training set in order to prioritize the important instances. One common pre-processing solution involves generating synthetic instances of the minority class, thereby balancing the training set and enhancing the performance specifically for the under-represented class.

Addressing this challenge requires the usage of specific techniques at different stages of the ML pipeline, categorized as pre-processing, special-purpose, and post-processing methods [12]. Pre-processing methods, which operate before the model training phase, include strategies such as data balancing, which tackles class imbalance by undersampling the majority class and oversampling the minority class. Special purpose methods, which directly influence the training process, include strategies like cost-sensitive learning. Post-processing methods, which are applied after the model training phase, encompass techniques like threshold moving.

Various data pre-processing techniques have been explored to address the class imbalance, which involve either augmenting the minority class or reducing instances of the majority class. One popular approach is the generation of synthetic examples to create new instances for the minority class. Strategies like SMOTE [18], Borderline-SMOTE [48], and ADASYN [49] have been widely utilized to tackle the imbalance problem. These methods utilize existing instances from the minority class, applying interpolation techniques to generate new instances and potentially introducing a specific bias during the generation process based on the chosen approach. Additionally, in recent times, the use of GANs has emerged as a viable approach for oversampling, which we will explore further next.

In numerous real-world scenarios, including cybersecurity, several challenges arise simultaneously [42, 143]. It's crucial that the solutions devised can tackle these overlapping challenges at once to ensure their effectiveness. In Chapter 3, we delve into a specific pre-processing technique called oversampling of the minority class using GANs. Our motivation stemmed from the observed gap in literature; there hasn't been a comprehensive study on the use of GANs for oversampling in various complex data situations. We discuss this in-depth in Chapter 3.

In the next section, we delve into the details of a GAN.

2.2 Generative Adversarial Networks

Since their introduction in 2014, GANs have made a significant impact on various ML tasks across multiple domains. Notably, GANs have demonstrated their effectiveness in image generation, as evidenced by numerous research papers [13, 59–62]. They have also proven valuable in image-to-image translation tasks [28, 55, 156], as well as image super-resolution [71]. In Chapter 3, we utilize GANs as a tool for oversampling, while in Chapter 4, we introduce a novel methodology for training GANs.

A basic GAN consists of two differentiable functions, namely the generator and the discriminator. The generator takes a sample from a multivariate distribution as input and maps it to a sample from another distribution. On the other hand, the discriminator's role is to distinguish the synthetic samples generated by the generator from real samples from the dataset. During each iteration of the learning process, the generator aims to deceive the discriminator by producing outputs that are accepted as real data, while the discriminator strives to accurately distinguish between real and fake samples. As they are trained together, the generator and discriminator engage in an adversarial competition, which can be explained using the game theory language. If the training process is deemed successful, we assert that the generator has effectively acquired knowledge of the dataset's distribution. As a result, it becomes capable of generating samples drawn from the learned distribution.

In order to learn the underlying distribution p_g of data x, we establish a prior on input noise variables $p_z(z)$. This allows us to map the noise space to the data space using a differentiable function $G(z;\theta_g)$, where G is implemented as a DL model with parameters θ_g. Additionally, we define a second DL model $D(x;\theta_d)$ that generates a scalar value. The output $D(x)$ represents the probability that x is a real data sample rather than generated by p_g [37].

The training process involves maximizing D to correctly classify both real training examples and samples generated by G. Simultaneously, G is trained to minimize the expression $log(1 - D(G(z)))$. Essentially, D and G engage in a two-player minimax game with the value function $V(G, D)$ [37]. The minimax objective function of a GAN is shown in Equation 2.1.

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}z \sim p_z(z)[1 - \log D(G(z))] \tag{2.1}$$

When trained on image datasets, the generator gradually improves its performance over many iterations until it can generate synthetic samples that closely resemble the

actual images. In other words, the generator learns to generate samples that follow the distribution of the dataset.

The basic form of a GAN, known as a vanilla GAN, operates as an unsupervised model, meaning it does not require class labels for the training examples. CGAN, introduced by [85], is a variant of GAN that require class labels during the training process. Unlike a vanilla GAN, where the generator learns the underlying distribution of the training dataset without class labels, a CGAN incorporates labels as additional input to the generator. This allows the generator in a CGAN to learn the distribution in a supervised manner, utilizing both the features and labels of the training examples. Figure 2.1 illustrates the architectures of both a GAN and a CGAN. In a GAN, the generator solely takes random noise as input, typically sampled from a multivariate normal distribution, resulting in no control over the generated examples' class. Conversely, in a CGAN, the generator receives both random noise and a label class as input, enabling it to generate samples specific to the given class. Consequently, the discriminator in a CGAN is exposed to examples with or without class labels. Equation 2.2 shows the objective function of a CGAN:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z|y))] \tag{2.2}$$

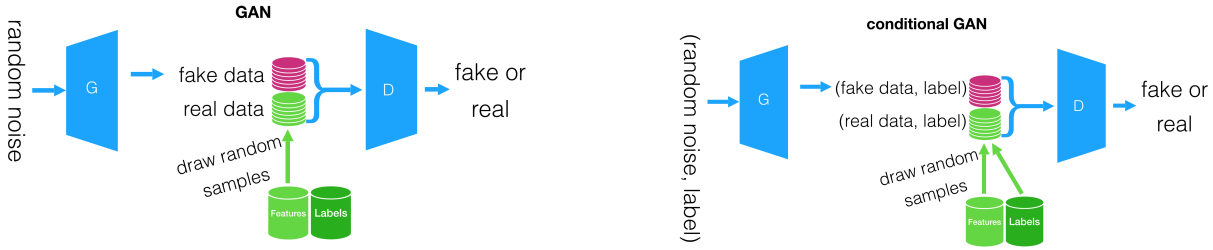

Figure 2.1: A snapshot of the training process of a GAN (left) and of a CGAN (right).

As mentioned earlier, Chapter 3 explores how well GANs tackle class imbalance in data difficulty factors. Furthermore, our training a CGAN on tabular datasets presented a challenge as they don't permit visual quality assessment during CGAN training. Chapter 4 proposes a solution for both dataset types.

2.3 Face Verification

Several strategies deceive FV systems: Dodging Attack alters face images unnoticeable to humans but confuses the system [16, 65]. Impersonation Attack tweaks one's image to mimic another for the system [101]. Our study unveils more FV vulnerabilities and introduces a superior attack framework. In Chapter 5 of this thesis, we delve into FV systems and propose a challenging strategy. This section further explores the concept of Biometric Verification in depth.

Biometric Verification involves the comparison of a user's biometric input, such as a fingerprint, with the biometric data of the user under scrutiny to determine if they match, indicating the same identity [107]. In the context of FV, directly comparing pixel-level details between two face images is impractical due to significant variations caused by factors like lighting conditions, different angles, and the presence or absence of glasses. This leads to the exploration of creating a robust representation of facial images that remains invariant to such changes, enabling effective face verification.

Deep neural networks have proven to be powerful tools for representation learning. One approach to conduct FV is training a deep neural network to classify different face images of various individuals. The intermediate layer of the deep neural network before the final classification layer serves as a meaningful representation of faces for the verification task. DeepFace [132], an early attempt at FV, follows this methodology. The process involves obtaining an alternative representation of face images by leveraging a pre-trained classifier on face images. Subsequently, these representations are compared using a distance metric such as Euclidean Distance. The deep neural network effectively maps each face image into a space that we call the embedding space, which serves as a superior representation for FV compared to the original image space. In this embedding space, when two points are close to each other (e.g. in terms of Euclidean distance), it indicates a high likelihood that the corresponding face images belong to the same identity.

Let us delve further into the embedding space of a deep neural network and examine the factors that contribute to its features and structure. In addition to the available dataset, the architecture of the deep neural network and the employed loss function during training play pivotal roles in shaping the embedding space. The development of deep neural networks has brought forth numerous novel ideas and architectures aimed at enhancing network performance. Notably, the introduction of Convolutional Neural Networks by Yann LeCun has had a profound impact on image-related tasks [69]. To address the issue of overfitting and facilitate effective training, the invention of dropout layers has significantly contributed [53]. The problem of vanishing or exploding gradients previously

hindered the training of very deep neural networks. InceptionNet, introduced by Szegedy et al., tackled this problem by implementing multiple loss functions within the network at different depths [130]. It also introduced Inception modules, a meticulously arranged set of CNNs that enhanced performance. The invention of Residual connections by He et al. effectively dealt with the vanishing or exploding gradients, enabling the training of even deeper neural networks and resulting in improved data representations for ML tasks [50]. In Chapter 5, our work utilizes a variation of FaceNet, which incorporates a mixture of Inception blocks from InceptionNet and Residual connections in its architecture, as described in [129] and implemented in https://github.com/davidsandberg/facenet/blob/master/src/models/inception_resnet_v1.py.

The choice of the loss function is a significant factor in shaping the embedding space. In the context of FV, a highly active research area focuses on proposing new and improved loss functions to enhance the representation of face images within the embedding space. Numerous research papers have made contributions in this area, introducing different loss functions to boost performance [22,23,75,116,124,127,128,139,141]. FaceNet, for instance, departs from the traditional softmax loss used for classification and introduces the Triplet Loss, which penalizes the network based on triplets of face images. The Triplet Loss aims to minimize the distance between an anchor and a positive sample of the same identity while maximizing the distance between the anchor and a negative sample of a different identity.

Chapter 3

Enhancing Predictive Modeling: Impact of Data Difficulty Factors on GAN-based Oversampling

In this chapter, we investigate the use of GANs to address different data challenges and examine their efficacy in improving classifiers under these challenging conditions. The work in this chapter is based on the following two papers:

- Ehsan Nazari, Paula Branco, and Guy-Vincent Jourdan. Using cgan to deal with class imbalance and small sample size in cybersecurity problems. In *2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE, 2021
- Ehsan Nazari and Paula Branco. On oversampling via generative adversarial networks under different data difficulty factors. In *Proceedings of the Third International Workshop on Learning with Imbalanced Domains: Theory and Applications*, volume 154 of *Proceedings of Machine Learning Research*, pages 76–89. PMLR, 17 Sep 2021.

3.1 Introduction

The performance of ML models is hindered by the class imbalance problem [18], which arises when the learning algorithm struggles to focus on the minority class due to the high frequency of the majority class. To address this problem, one of the viable solutions

is the implementation of pre-processing methods. Pre-processing methods act before the learning procedure by altering the distribution of training data. Examples of pre-processing methods include oversampling, undersampling, and a combination of both, which enable the learning algorithm to better handle minority class cases. In this study, we are mainly looking at pre-processing methods. This is because new ways have been developed to create synthetic data, and we think it is important to understand how these pre-processing methods work with these new data creation techniques.

In the domain of Cybersecurity applications, the class imbalance problem holds particular relevance and can have serious consequences [57]. For example, in tasks involving the detection of attacks or intrusion, the majority of the data represents normal behavior, while abnormal cases make up only a small percentage. The timely detection of these abnormal cases is crucial, but their low representation in the available dataset can hamper the performance of the learning algorithm.

In addition to the class imbalance problem, various other data characteristics have been shown to significantly affect the performance of learners in real-world scenarios [11, 14, 77]. Factors such as class overlap and small sample size pose challenges and often coexist with the imbalance problem. The recognition of the minority class is particularly compromised when these factors are present in imbalanced datasets [89]. However, the impact of these data difficulty factors on the application of oversampling using GAN has not been investigated thoroughly.

The primary objective of this chapter is to investigate how various data difficulty factors found in imbalanced datasets affect the performance of GAN-based oversampling strategies. The motivation behind centering our attention on GANs lies in their growing significance in diverse research fields, coupled with their inherent capacity to generate synthetic samples, making them a compelling choice for addressing data imbalance challenges. This chapter aims to provide a structured framework to discern the strengths and vulnerabilities of GAN-based oversampling approach. While our experiments focuses on the capabilities of CGAN for data augmentation, the framework presented can be generalized and applied to other oversampling techniques as well.

Our analysis will delve into the effects of multiple data difficulty factors on two dataset categories: imagery and Cybersecurity. Specifically, for imagery datasets, the focus is on understanding the influence of factors such as imbalance ratio, sample size, data dimensionality, and class overlap when utilizing a CGAN architecture for data augmentation. The experiments conducted employ a well-known image dataset (MNIST) [70] to generate multiple versions with varying levels of the aforementioned data difficulty factors, allowing for controlled observation of their effects. For Cybersecurity data tests, the emphasis lies

in comprehending the impact of the imbalance ratio and sample size on performance.

This chapter provides three primary contributions:

- provides a comprehensive experimental investigation into the impact of each data difficulty factor and their combined effects on CGAN performance for MNIST dataset;
- provides an extensive analysis of the use of CGAN as an oversampling method for Cybersecurity datasets;
- establishes guidelines on the effectiveness of this oversampling method.

This chapter is structured as follows: Section 3.2 reviews related studies. Section 3.3 details our approach to tackle class imbalance in binary classification using CGANs. Section 3.4 covers the experimental procedures, datasets, and settings. The results are then presented in Section 3.5, beginning with image dataset findings and followed by results from Cybersecurity tabular datasets. The chapter concludes in Section 3.6, and possible future research directions are proposed in Section 3.7.

3.2 Related Work

In this section, we will introduce CGAN. Then, we will discuss the various applications of CGAN in the domain of Cybersecurity and review related works that focus on addressing the challenge of class imbalance.

3.2.1 Applications of GANs in Class Imbalance Tasks

A plethora of GAN-based solutions have been successfully employed in various computer vision tasks, such as image translation, super-resolution, synthesis, and video generation [96]. GANs have also been utilized for classification in imbalanced image datasets, with approaches categorized into classification, object detection, and segmentation tasks [111]. Comprehensive surveys by [96, 111] provide detailed insights into these developments.

In contrast, the application of GAN in imbalanced non-imagery datasets has received less attention. In one study [30], a vanilla GAN was trained solely on the minority class and applied to address credit card fraud detection. Another study [29] explored a variant of CGAN to oversample the minority class in tabular data with categorical variables.

Furthermore, in the domain of Natural Language Processing, SeqGAN was introduced in [149], a framework for sequence generation that incorporates reinforcement learning into the generator.

3.2.2 Applications of GANs in Cybersecurity Domain

GAN has found numerous applications in the field of Cybersecurity. One notable application involves the generation of attacks targeting neural networks. Additionally, GAN has been utilized for mitigating adversarial attacks, improving the accuracy of malware and fraud detection systems, and addressing concerns related to data privacy.

Adversarial attacks are techniques designed to deceive discriminative neural network models by causing them to misclassify input data. The primary objective is to modify the original image in a manner that appears nearly identical to the human eye while inducing the classifier to predict an incorrect class. This is achieved by introducing small perturbations to the input image. It is widely recognized that neural network classifiers are vulnerable to adversarial attacks [103]. Several GAN architectures have been employed to generate such image perturbations, as demonstrated in studies such as [7,125,142,144,152].

The concept of adversarial attacks in images has also been applied to malware datasets. In a specific example, a GAN-based approach was introduced in [117] that can successfully deceive malware detectors with up to 99% of the generated adversarial examples.

In the context of using GAN as defense mechanisms against adversarial attacks, various GAN architectures have been introduced to protect classifiers from being vulnerable to such attacks (e.g., [72,74,119]). An example is presented in [110], where a specific type of GAN known as defenseGAN serves as a filter. It takes the attacked image as input and produces a distilled image, effectively resisting the adversarial attack.

GAN has found applications in improving the performance of malware and fraud detectors as well. For instance, in [58], a GAN is employed to classify and detect malware, while in [153], a GAN is utilized to enhance the capabilities of a fraud detector.

In order to tackle privacy concerns associated with the transfer of sensitive data, various approaches (e.g., [15, 19, 64, 73, 98, 135, 146]) have utilized GAN. These solutions involve training GAN to learn the underlying distribution of the sensitive data, allowing them to generate entirely new synthetic datasets. By utilizing these synthetic datasets, it becomes possible to work with the data without being subject to the constraints imposed on the original dataset.

The potential of utilizing GAN to address class imbalance remains largely unexplored and not thoroughly understood, as highlighted in a survey [112]. Additionally, the coexistence of other data difficulty factors, which commonly occur in real-world imbalanced problems, has yet to be investigated.

Our objective is to evaluate the performance of a CGAN under different conditions that can impede its effectiveness in binary imbalanced learning tasks. Specifically, we will examine the impact of the following factors on imagery datasets: (1) varying numbers of features per sample, (2) different levels of class overlap, (3) different imbalance ratios, and (4) different sample sizes. Furthermore, we aim to extend our experiments to Cybersecurity datasets, as we have observed a lack of research on the application of GANs to address the intersection of class imbalance and small sample issues in the Cybersecurity domain.

3.3 CGAN-based Data Generation for Imbalanced Datasets

To evaluate the oversampling framework we use a 5-fold cross-validation process. Two settings are used for evaluation: the baseline setting, shown in Figure 3.1 and the CGAN-based augmentative setting, shown in Figure 3.2. We evaluate and test the CGAN-based augmentative framework on both imagery datasets and various Cybersecurity datasets that are in a non-imagery (tabular) format. Additionally, we generate different versions of these datasets with different characteristics that we aim to analyze.

One challenge in our setting is that in tabular non-imagery datasets, it is not possible to visually assess the results obtained from the outputs of the CGAN generator.[1]

The baseline setting, depicted in Figure 3.1, involves a 5-fold cross-validation approach where a model is trained and tested using the original imbalanced dataset.

The second setting, referred to as the CGAN-based augmentative oversampling strategy and illustrated in Figure 3.2, introduces two additional steps to the baseline approach. In these steps (II and III, with green background), a CGAN architecture is trained using the original imbalanced training set, incorporating both minority and majority class cases. Following this training process, the trained generator is utilized to generate synthetic data for the minority class. This synthetic data is then merged with the original training set, resulting in a modified balanced training set.

[1] This challenge serves as the basis for the subsequent chapter of the thesis. In Chapter 4, we will present a method that effectively addresses the raised issue by providing an efficient and convenient approach to training a GAN on both imagery and tabular data.

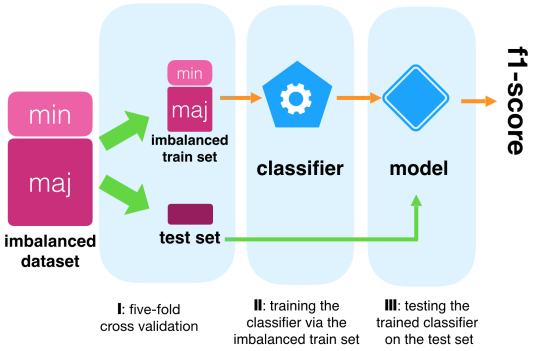

Figure 3.1: The baseline configuration used to evaluate the performance of a classifier on an imbalanced dataset.

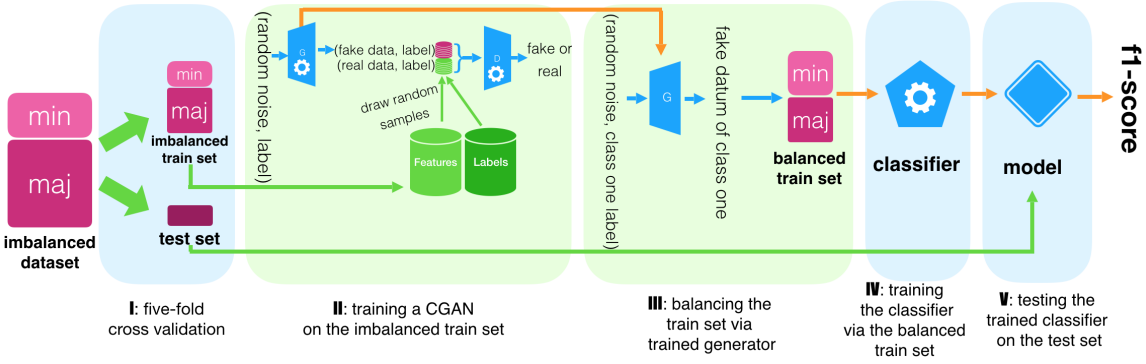

Figure 3.2: The experimental setup for evaluating the performance of a classifier utilizing the CGAN-based oversampling method.

Performance assessment metrics, specifically the F1-score in the context of imbalanced datasets, are computed at the conclusion of each setting. For a more comprehensive understanding of the baseline framework and the CGAN-based oversampling architecture, refer to Figure 3.1 which illustrates the scenario where the classifier is presented with the original imbalanced training data, and Figure 3.2 which depicts the situation where the classifier receives a balanced training dataset achieved by augmenting the underrepresented class using a CGAN.

3.4 Experimental Evaluation

This section provides the details of the imagery and tabular datasets used and the experimental settings of our empirical evaluation.

3.4.1 Imagery Datasets

For the initial phase of our experiments, we opted to use the MNIST dataset [70]. This dataset comprises gray-scale images of hand-written digits, each measuring 28 pixels by 28 pixels. Our objective is to create multiple datasets with varying levels of the following attributes: (i) sample size; (ii) class overlap; (iii) imbalance ratio; and (iv) number of features. This approach enables us to manipulate specific challenging factors in the data for our study. Now, we will outline the procedure we employed to generate the data and ensure the presence of these characteristics in each dataset.

The initial training set of the MNIST dataset includes a total of 5842 occurrences of the digit 'four'. To form all the generated datasets, we chose the instances of the digit 'four' from MNIST as the primary majority class cases. The sample size of each dataset was determined based on the majority class. To maintain the original order of the 5842 instances of the digit 'four', we specifically selected the first 1000, 400, 200, and 100 instances from this class. These sets will serve as the majority class cases for four foundational datasets, each varying in terms of sample size. Later on, we will combine these datasets with the minority class examples using different combinations of attributes, which will be explained in detail below.

To regulate the extent of class overlap, we implemented a transformation on the initial majority class images to generate the minority class. To achieve varying levels of class overlap, we introduced three degrees of rotation to the original 'four' digits. Consequently, our minority class instances were derived by rotating the original majority class image

by 30 degrees, 45 degrees, or 90 degrees. Essentially, the task for the classifier is to distinguish between the upright 'four' digits and 'four' digits after they have been rotated. The underlying assumption is that when the minority class images are produced through slight rotations of the majority class images, it becomes more challenging for a classifier to differentiate between them. For instance, employing the original images and their rotations of 30 degrees yields a higher degree of class overlap compared to using rotations of 90 degrees. Therefore, discriminating between a regular 'four' and a 90-degree rotated image is comparatively easier, while distinguishing the original image from a 45-degree rotated image is more difficult. Among the three, the binary classification task involving a normal 'four' and a 30-degree rotated image poses the greatest difficulty. Overall, this method offers a controlled way to introduce class overlap and varying degrees of recognition difficulty, without the added complexity of different digit shapes. It provides a clear, focused challenge for the classifier, enabling a straightforward evaluation of its performance on different variations of class overlap.

The rotation of the images was executed using the default settings of the PIL library in Python. Examples of the minority class instances generated through different rotations can be observed in Figure 3.3. In this figure, the left column displays a single majority class instance represented with varying numbers of features. The process of obtaining the majority class instances will be elaborated below. The other three columns exhibit the corresponding generated minority class instances for the defined rotations.

Additionally, we generated datasets with varying degrees of class imbalance. For this set of experiments, we adopted the definition of imbalance ratio (IR) as the quotient of the number of minority class images divided by the number of majority class images. We applied three imbalance ratios (0.4, 0.2, 0.1) specifically to the minority class. To achieve this, we calculated the total number of majority class instances and multiplied it by the selected imbalance ratio, resulting in the total number of minority class instances to be included in the dataset. This manipulation of data difficulty was implemented to investigate the relationship between the imbalance ratio and the CGAN's capacity to oversample the minority class.

Lastly, we generated datasets with varying numbers of features by resizing the original 28×28 images to smaller dimensions. Consequently, for each dataset, we produced resized versions of the images with dimensions of 14×14, 8×8, and 4×4 pixels. The resizing process was accomplished using the open-cv library in Python, employing the INTER AREA interpolation algorithm. The primary objective behind reducing the size of the original images was to obtain class representations with a reduced number of features. As a result, the described procedure allowed us to generate datasets with 784, 196, 64, and 16 features. By examining this data difficulty aspect, we aim to gain insights into the impact

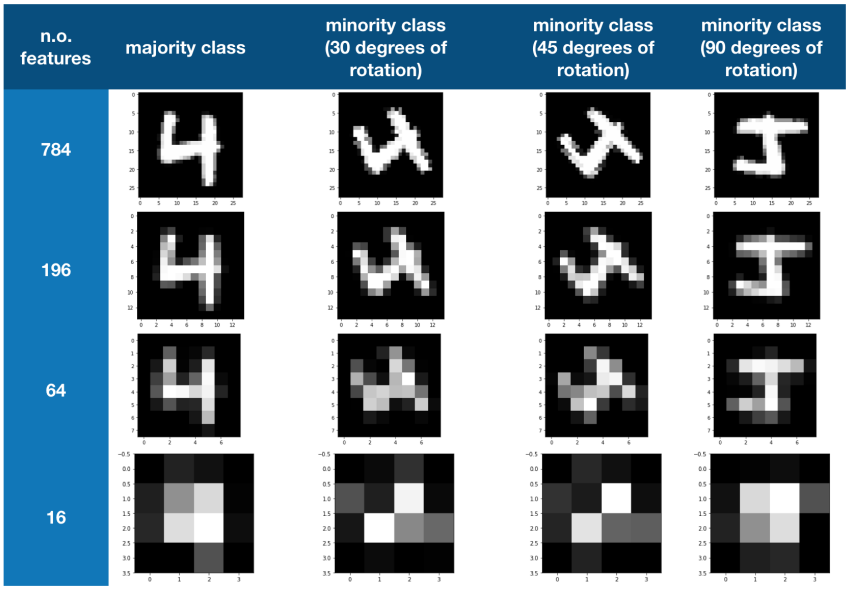

Figure 3.3: Demonstration of the creation of diverse image sizes and rotations applied to an initial MNIST dataset image located in the top left corner.

of feature count on the effectiveness of the CGAN in addressing the imbalance issue. We hypothesize that a significant reduction in the number of available features for the CGAN could present a major challenge. Examples of the minority class instances generated at different sizes can be observed in Figure 3.3.

The settings employed to generate each of the data difficulty characteristics are summarized in Table 3.1. In total, we created 144 different datasets by considering all possible combinations of configurations for each characteristic (4 for sample size, 3 for class overlap, 3 for imbalance ratio, and 4 for the number of features). To ensure the reproducibility of our research, all the generated datasets can be accessed freely at https://github.com/enazari/GAN-upsampling-LIDTA21.

3.4.2 Tabular Cybersecurity Datasets

Non-imagery Cybersecurity datasets were also used in our experiments. These datasets are derived from four publicly accessible datasets: two binary and two multi-class datasets. Since our objective is to evaluate an oversampling framework for binary classification problems, we converted the multi-class datasets into binary tasks by selecting two classes from

Table 3.1: An overview of the values and concise explanations for the various data difficulty factors utilized in imagery datasets.

Data Difficulty Factor	values used	explanation
Sample Size	$\{1000; 400; 200; 100\}$	Set by fixing the majority class cases used in each dataset
Class Overlap	$\{30°; 45°; 90°\}$	Rotation applied to generate the minority class cases
IR	$\{0.4; 0.2; 0.1\}$	Imbalance ratio obtained by adding or removing minority class cases
Nr. of Features	$\{28 \times 28; 14 \times 14; 8 \times 8; 4 \times 4\}$	Image resize for all cases

each dataset. As a result, we obtained a total of seven binary classification problems. Figure 3.4 illustrates a tree-like diagram representing these datasets.

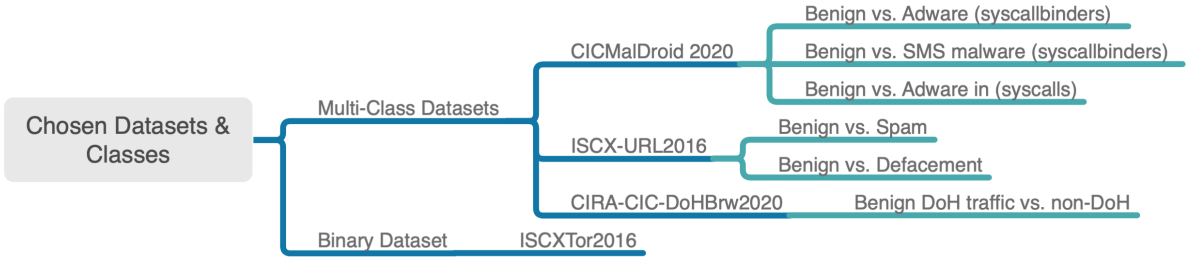

Figure 3.4: Visualization depicting the base Cybersecurity datasets utilized to generate 114 derived datasets.

To effectively examine the impact of varying class imbalance and small sample sizes, we implemented additional modifications to ensure control over these characteristics from the outset. Each dataset generated in our repository is defined based on two key features: (i) the count of the majority class, representing the total number of instances belonging to the majority class (this count depends on the available majority class examples in the base dataset); and (ii) the imbalance ratio, defined as the ratio between the number of minority class instances and the number of majority class instances. Through a process outlined below, we derived a total of 114 different datasets for our experimental analysis.

Our primary approach for generating datasets with specific characteristics follows these steps. Firstly, we randomly choose the desired number of majority class instances from the available data. Subsequently, we randomly select the corresponding number of minority class instances to be included in the dataset, ensuring that the desired imbalance ratio is achieved. This process grants us complete control over both characteristics. It is important

to note that the number of majority class instances may differ across base datasets, as it is constrained by the initial availability of majority class examples in each dataset.

The "cira" base dataset [86] employs a two-layered approach to capture benign and malicious DNS-over-HTTPS (DoH) traffic, as well as non-DoH traffic. For our experiments, we exclusively utilize the benign DoH traffic and non-DoH classes from this dataset. To explore the effects of varying class imbalance and sample size, we derive a total of 24 datasets from this base dataset. These datasets are created by considering eight different majority class counts (specifically, 100, 200, 500, 1000, 2000, 4000, 8000, and 16000) and employing three levels of imbalance ratio (0.1, 0.2, and 0.4).

The base dataset referenced as "tor" [46] consists of time-based features representing TOR traffic and non-TOR traffic in a binary classification setting. From this base dataset, we derive a total of 21 datasets. These derived datasets are generated by considering three levels of imbalance ratio (0.1, 0.2, and 0.4) and incorporating seven different majority class counts (specifically, 100, 200, 500, 1000, 2000, 4000, and 8000).

We also incorporate the Android malware dataset [82] in our research, which encompasses five target classes: Adware, Banking, SMS malware, Riskware, and Benign. This dataset consists of various features extracted from 11,598 APK files. The first set of features includes frequencies of system calls, binders, and composite behaviors, while the second set focuses solely on frequencies of system calls [82]. For our experiments, we utilize these two sets of features, which correspond to two distinct base datasets. In this study, we refer to the dataset with 470 features as "cic-syscallsbinders", and the dataset with 139 features as "cic-syscalls".

In order to convert these datasets into binary classification problems, additional processing was required. For the "cic-syscallsbinders" dataset, we selected the benign class and the adware class to create a base dataset named "cic-syscallsbinders-adware". From this base dataset, we generated a total of 12 datasets by incorporating three different levels of imbalance ratio (0.1, 0.2, and 0.4) and considering five different majority class counts of 100, 200, 500, and 1000.

Similarly, we utilized the benign class and the SMS adware class from the "cic-syscallsbinders" dataset to construct the base dataset "cic-syscallsbinders-smsadware". From this base dataset, we derived 12 datasets using the same three levels of imbalance ratio (0.1, 0.2, and 0.4) and the same five different majority class counts of 100, 200, 500, and 1000.

Moreover, we selected the benign class and the adware class from the "cic-syscalls" dataset to form the base dataset called "cic-syscalls-adware". From this base dataset, we generated a total of 12 datasets by incorporating three levels of imbalance ratio (0.1, 0.2,

and 0.4) and considering five different majority class counts of 100, 200, 500, 1000, and 1500.

The paper [83] introduces a dataset containing benign and malicious URLs. This dataset encompasses various target classes, including Benign URLs, Spam URLs, Phishing URLs, Malware URLs, and Defacement URLs. For convenience, we refer to this dataset as "iscx". To transform the multi-class problem into binary classification tasks, we selected the benign class and a malicious class. This led to the creation of the base dataset "iscx-spam", which consists of examples from the benign and Spam URLs classes. From this base dataset, we generated a total of 15 datasets by three levels of imbalance ratio (0.1, 0.2, and 0.4) and considered five different majority class counts of 100, 200, 500, 1000, and 2500.

Similarly, the "iscx-defacement" dataset was obtained from the same "iscx" dataset by selecting the benign class and the Defacement URLs class. We derived 15 datasets from this base dataset by applying three levels of imbalance ratio (0.1, 0.2, and 0.4) and considering five different majority class counts of 100, 200, 500, 1000, and 2500.

Table 3.2 presents the varied characteristics applied to each base dataset, along with the corresponding number of generated datasets included in our repository. In total, we obtained and utilized 114 datasets with carefully controlled imbalance ratios and the number of majority class examples.

Table 3.2: Characteristics of the generated cybersecurity datasets and the architecture of the classifiers used, detailing base dataset (Base DS), imbalance ratio (IR), and number of generated datasets (Gen. DS).

Base DS	**IR**	**Majority Class Count**	**# Gen. DS**	**Classifier Architecture**
cira	0.1, 0.2, 0.4	100, 200, 500, 1000 2000, 4000, 8000, 16000	24	one hidden layer of 10 perceptrons
tor	0.1, 0.2, 0.4	100, 200, 500, 1000 2000, 4000, 8000	21	one hidden layer of 10 perceptrons
cic-syscallsbinders-adware	0.1, 0.2, 0.4	100, 200, 500, 1000	12	two hidden layers of 20 perceptrons
cic-syscallsbinders-smsadware	0.1, 0.2, 0.4	100, 200, 500, 1000	12	two hidden layers of 20 perceptrons
cic-syscalls-adware	0.1, 0.2, 0.4	100, 200, 500, 1000 1500	15	two hidden layers of 100 perceptrons
iscx-spam	0.1, 0.2, 0.4	100, 200, 500, 1000 2500	15	one hidden layer of 10 perceptrons
iscx-defacement	0.1, 0.2, 0.4	100, 200, 500, 1000 2500	15	two hidden layers of 100 perceptrons

3.4.3 Experimental Setting

Our objective is to evaluate the influence of the CGAN-based oversampling approach in both imagery and tabular Cybersecurity classification problems. We seek to examine the impact on performance across different scenarios characterized by varying data properties.

After generating the repository of derived datasets, consisting of 144 imagery datasets and 114 tabular Cybersecurity datasets, we proceeded with two tests: one utilizing the original training set and the other employing the CGAN-based oversampling technique to balance the training set prior to applying the learning algorithm. These two settings are presented in Figure 3.1 and Figure 3.2, respectively. Our focus was on evaluating the impact on the performance of both the minority and majority classes when training a classifier on imbalanced and balanced datasets using the CGAN oversampling method. Furthermore, we aimed to identify the specific situations in which the CGAN-based augmentative oversampling demonstrated greater efficiency. This involved evaluating the performance across different factors, including the imbalance ratios, sample sizes, class overlap, and number of features for imagery datasets, as well as the imbalance ratios and sample sizes for Cybersecurity datasets. In total, we conducted 288 tests on the 144 imagery datasets and 228 tests on the 114 Cybersecurity datasets.

We employed a 5-fold non-stratified cross-validation procedure to assess the F1-score (refer to Equation 3.1) of the majority and minority classes of the classifier in both the original training set and the balanced training set. When calculating the F1-score for the minority (majority) class, we considered the minority (majority) class as the positive class, i.e., the class of interest. This means that true positive (TP) cases are correctly predicted minority (majority) class cases, false positive (FP) cases are majority (minority) class cases incorrectly predicted as the minority (majority) class, and false negative (FN) cases represent the minority (majority) class cases incorrectly predicted as the majority (minority) class.

We assessed the F1-score for each class and determined the impact of CGAN-based oversampling. This metric evaluates the improvement in the F1-score achieved by balancing the training set. We also computed other auxiliary metrics to easily observe the performance changes (see Table 3.3). We compute the impact of the strategy by taking the ratio of the F1-score obtained from the balanced version of the dataset to the F1-score obtained from the original imbalanced version (metrics min_f1_imp and maj_f1_imp in Table 3.3). If no change (gains/losses) is observed, the impact is 1. A higher value indicates performance improvements resulting from balancing, while a lower value indicates a decrease in performance. For a comprehensive list of the metric notations and their definitions, please refer to Table 3.3.

$$\text{F1-score} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})} \tag{3.1}$$

Table 3.3: Notation used for reporting the results.

Notation	Description
maj_f1	F1-score of the majority class before oversampling
maj_f1_bal	F1-score of the majority class after oversampling
min_f1	F1-score of the minority class before oversampling
min_f1_bal	F1-score of the minority class after oversampling
min_f1_imp	impact on the minority class F1-score measured as the ratio of min_f1_bal to min_f1
maj_f1_imp	impact on the majority class F1-score measured as the ratio of maj_f1_bal to maj_f1
maj_count	number of majority class samples,which represents the sample size difficult factor
imbalance_ratio	the ratio between the number of minority class samples and the number of majority class samples
rot_angle	the angle between the minority class samples and the majority class samples
nr_features	the number of each sample's features

The CGAN architecture remains consistent throughout all experiments, except for the output layer of the generator and the input layer of the discriminator, which are adjusted to accommodate the specific number of features in each dataset. All CGAN models are trained for 1500 iterations on the derived datasets. We utilized the implementation of CGAN from the GitHub repository https://github.com/eriklindernoren/Keras-GAN/tree/master/cgan as our base architecture.

The classifier model utilized in our experiments is a multilayer perceptron. The input layer of the classifier is adjusted to match the number of features in each dataset. For imagery datasets, a single hidden layer with 10 neurons is employed. The number of hidden layers for Cybersecurity datasets was manually tuned based on preliminary experiments. A summary of the dataset generation settings for the 114 datasets derived from the base Cybersecurity datasets, along with the specifications of the classifiers used for each group, is presented in Table 3.2.

Table 3.4: The average F1-score results from 144 tests conducted both before and after applying CGAN data augmentation.

metric	score without augmentation	score with augmentation
F1-score of the majority class	**0.940310**	0.912214
F1-score of the minority class	0.434325	**0.701059**

3.5 Results and Discussion

3.5.1 Imagery Datasets Results and Discussion

Overall Results

Our initial analysis provides a summary of our findings. Table 3.4 presents the performance results obtained from using the original imbalanced datasets and applying CGAN as a data augmentation technique. It is observed that the aggregated F1-scores after data augmentation exhibit a 3% decrease for the majority class and a notable 26% increase for the minority class. This suggests that employing CGAN-based augmentative oversampling offers an advantage for datasets with multiple data difficulty factors, particularly for the minority class. However, a slight decrease in F1-score for the majority class is expected when using data augmentation techniques.

Results by Rotation Angle

Table 3.5 provides an overview of the results obtained when considering different levels of class overlap through rotated images. It can be observed that all metrics calculated for both the minority and majority classes exhibit lower scores for lower rotation angles. This aligns with our initial expectations, as lower rotation angles lead to images with higher overlap, thus increasing the difficulty of the classification task.

Furthermore, we observe an increase in the F1-score for the minority class after applying the CGAN oversampling method, while the F1-score for the majority class shows only a marginal decrease. This demonstrates the effectiveness of our method in addressing imbalanced problems with class overlap. The improvements achieved for the minority class are particularly significant, ranging from 157% to 166%. These gains are observed across all rotation angles, with higher gains observed for lower rotation angles, which correspond to scenarios with greater class overlap.

Table 3.5: Average results obtained from all tests conducted based on rotation angle.

rot_angle	maj_f1	maj_f1_bal	maj_f1_imp	min_f1	min_f1_bal	min_f1_imp
30	0.930060	0.883843	0.95	0.358458	0.596278	1.66
45	0.942688	0.923152	0.98	0.450307	0.732212	1.63
90	0.948183	0.929648	0.98	0.494209	0.774687	1.57

Table 3.6: Average results obtained from all tests conducted based on the number of features.

nr_features	maj_f1	maj_f1_bal	maj_f1_imp	min_f1	min_f1_bal	min_f1_imp
16	0.900711	0.794932	0.88	0.045959	0.471036	10.25
64	0.935136	0.926155	0.99	0.380603	0.717461	1.89
196	0.957646	0.960770	1.00	0.601239	0.788395	1.31
784	0.967748	0.966999	1.00	0.709498	0.827345	1.17

Results by Number of Features

The performance results for both classes, aggregated by the number of features of the datasets, are presented in Table 3.6. The findings clearly demonstrate a positive impact of the applied augmentative strategy. Remarkably, this impact is most significant for datasets with the most challenging conditions, showcasing an improvement of 1025% compared to the original imbalanced dataset for cases with 16 features. This observation is particularly noteworthy as it highlights the effectiveness of CGAN in datasets with a limited number of available features.

Results by Imbalance Ratio

The impact of different imbalance ratios on performance is shown in Table 3.7. Similar to the previously mentioned data difficulty factors, there are significant improvements in the results of the minority class when CGAN data augmentation is applied. However, there is a slight decrease in the F1-score for the majority class. Additionally, we observe that the oversampling strategy provides the greatest benefits to the minority class in the most challenging scenarios characterized by a larger imbalance between the minority and majority class cases. It is worth noting that although there is some performance loss in the majority class, the magnitude of this loss is much smaller compared to the magnitude of the observed gains.

Table 3.7: Average results obtained from all tests conducted based on imbalance ratio.

imbalance_ratio	maj_f1	maj_f1_bal	maj_f1_imp	min_f1	min_f1_bal	min_f1_imp
0.1	0.961613	0.921095	0.96	0.268097	0.586425	2.19
0.2	0.940043	0.907120	0.96	0.414639	0.707478	1.71
0.4	0.919275	0.908428	0.99	0.620238	0.809274	1.30

Table 3.8: Average results obtained from all tests conducted based on majority class count.

maj_count	maj_f1	maj_f1_bal	maj_f1_imp	min_f1	min_f1_bal	min_f1_imp
100	0.916672	0.818838	0.89	0.189529	0.548599	2.89
200	0.934167	0.926713	0.99	0.359792	0.662593	1.84
400	0.947295	0.944452	1.00	0.498504	0.765536	1.54
1000	0.963108	0.958853	1.00	0.689474	0.827510	1.20

Results by Sample Size

Table 3.8 presents the comprehensive results obtained for different sample size scenarios. We observe similar improvements in the results, indicating that CGAN is effective in dealing with this data difficulty factor as well. Furthermore, it is evident that CGAN provides greater performance gains for the most challenging scenarios.

Results by Four Factors

Figure 3.5 displays the F1-score results for the minority class across the four challenging factors investigated in this study: class overlap (represented by rotation angles), number of features (indicated by image resizing), imbalance ratio (determined by the addition of minority class examples), and sample size (determined by the base majority class count). The figure demonstrates that in all cases, utilizing CGAN for data balancing positively impacts the learner's performance. Furthermore, it is evident that this method offers significant advantages even for the most difficult tasks, where the original imbalanced datasets exhibit lower performance results.

Results by Combination of Two Factors

We conducted an analysis to assess the impact of multiple data difficulty factors on performance by considering all possible combinations of two factors. Figure 3.6 presents the F1-score of the minority class for these combinations, with the factors arranged in descending

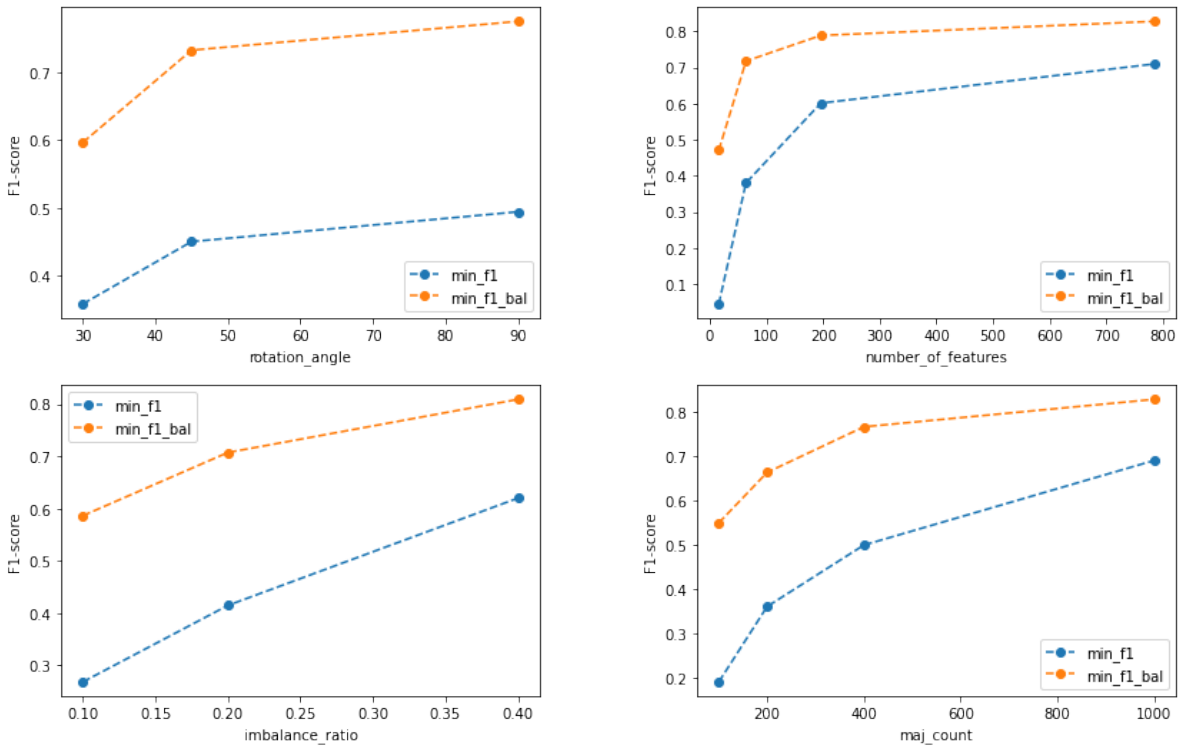

Figure 3.5: The F1-score of the minority class, represented by the blue line, is shown before and after oversampling using CGAN in the following four categories: rotation angle (top left), number of features (top right), imbalance ratio (bottom left), and sample size (bottom right). The orange line represents the F1-score after the application of oversampling.

order of difficulty. We observe that as the combination of factors becomes less challenging, the performance of the minority class improves, both with and without the application of CGAN oversampling. Additionally, we notice that the performance gains achieved with CGAN tend to be smaller as the difficulty of the factors decreases. The toothed saw observed in most combinations of factors for the dataset modified using CGAN indicates that each difficulty factor has a significant impact when the other considered factor remains constant.

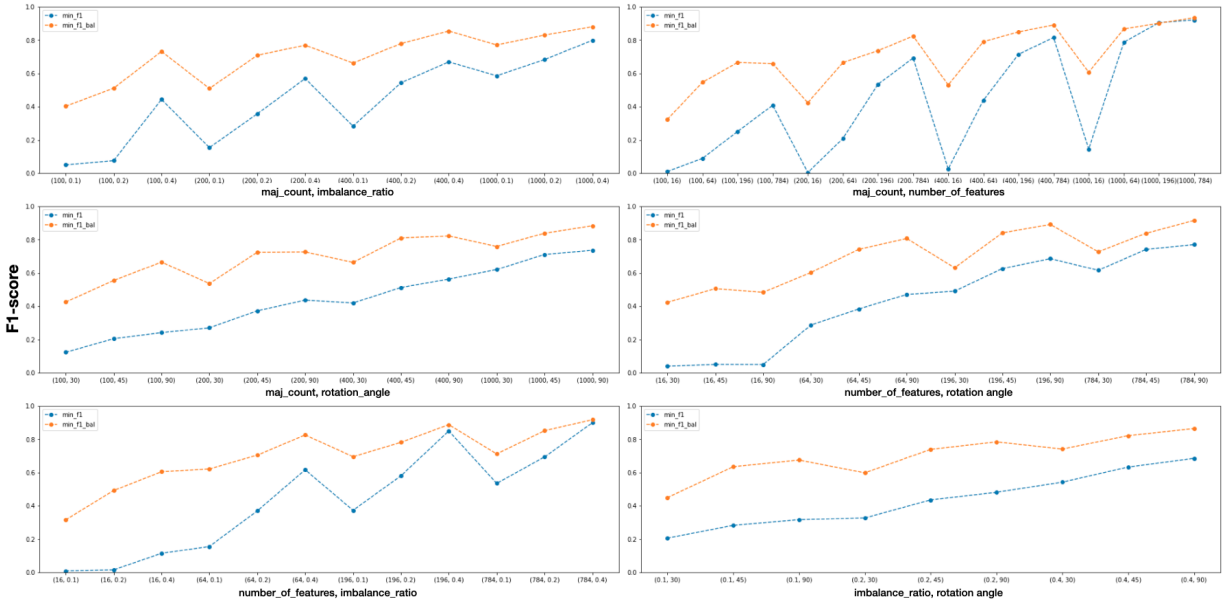

Figure 3.6: F1-score of the minority class is shown for two data difficulty factors, with blue indicating the original data and orange indicating the oversampled data. The combinations of factors, from top left to bottom right, are as follows: sample size-imbalance ratio, sample size-number of features, sample size-class overlap, number of features-class overlap, number of features-imbalance ratio, and imbalance ratio-class overlap.

3.5.2 Cybersecurity Datasets Results and Discussion

Overall Results

Table 3.9 displays the average results of all experiments grouped by the base datasets. For instance, if we consider the "tor" dataset, which has 21 derived datasets, the results in the

"tor" row represent the average scores obtained from tests conducted on these 21 datasets, considering different imbalance ratios and majority class counts.

Overall, significant improvements in the min_f1 performance are observed when utilizing CGAN-based oversampling. The min_f1_imp value consistently exceeds 1, indicating positive gains compared to the original imbalanced dataset. However, the extent of improvement varies across datasets. For example, the "cic-syscallsbinders-adware" dataset shows a substantial enhancement of 15.41 times compared to the original and not-oversampled dataset, while the gains in "icsx-defacement" are minimal (1.01). It is important to note that these results represent averages across different imbalance ratios and sample sizes for each dataset. Nevertheless, the variation in gains is evident. For the "icsx-defacement" base dataset tests where the gains are minimal, we speculate that it could be attributed to initially high performance in the minority class, leaving less room for improvement.

Table 3.9 also reveals a decline in the overall performance of the majority class. This outcome is anticipated when employing oversampling methods, as we typically achieve considerable enhancements in the minority class at the expense of some deterioration in the majority class performance. The datasets exhibiting larger improvements in the minority class also tend to experience a greater decrease in the majority class performance.

Table 3.9: Average results across all imbalance ratios and majority class counts for each dataset.

dataset	min_f1	min_f1_bal	min_f1_imp	maj_f1	maj_f1_bal	maj_f1_imp
cic-syscalls-adware	0.029981	0.267623	8.93	0.897190	0.797150	0.89
cic-syscallsbinders-adware	0.018313	0.282120	15.41	0.897921	0.744729	0.83
cic-syscallsbinders-smsmalware	0.194751	0.539318	2.77	0.915258	0.842134	0.92
cira	0.695983	0.883897	1.27	0.968960	0.970149	1.00
iscx-defacement	0.801138	0.807846	1.01	0.966230	0.961493	1.00
iscx-spam	0.605218	0.783269	1.29	0.958686	0.951434	0.99
tor	0.448913	0.694139	1.55	0.923881	0.919373	1.00

Results by Sample Size

Table 3.10 presents the results categorized by dataset and majority class count. The results are averaged across all tested imbalance ratios. The findings indicate a clear trend in the min_f1_imp values: as the number of samples increases, the performance gains diminish. This implies that the CGAN-based augmentative oversampling method is most effective

when the available sample size is small, resulting in more significant improvements in such scenarios. Notably, even with smaller sample sizes, the CGAN is capable of learning the underlying data distribution and generating meaningful synthetic examples. However, as more samples become available, the classifier's effectiveness increases, leading to a reduced impact of the oversampling strategy. In instances where the min_f1_imp values are close to 1, indicating minimal gains, it is observed that the base min_f1 on the original imbalanced data is already high. Conversely, in cases where the min_f1 values are low, remarkable improvements are achieved through CGAN augmentative oversampling. For instance, notable improvements can be seen in the "cic-syscalls-adware" dataset with 100 and 200 majority class counts, where the performance changes from zero to 0.22 and 0.24, respectively. As for the maj_f1_imp results, they are around 1, indicating that CGAN oversampling has little to no impact on the F1-score of the majority class.

Results by imbalance ratio

Table 3.11 displays the average results for each base dataset and imbalance ratio, aggregated based on the majority class count. It can be observed that as the imbalance ratio increases, the improvement in min_f1_imp decreases. Similarly, the F1-score of the minority class on the original data (min_f1) tends to increase as the imbalance ratio increases. In cases where the classifier performs well on the original dataset, the room for improvement through oversampling shrinks.

Figure 3.7 illustrates the average F1-score of the minority class, represented by the blue dashed line for the original data and the orange dashed line for the oversampled data, across different base datasets and imbalance ratios. In most cases, the F1-score of the minority class after data augmentation is higher than the F1-score before augmentation, indicating the clear advantage of this technique. However, for the "iscx-defacement" dataset, the gains from applying CGAN-based oversampling are not evident. It should be noted that in this particular case, the classifier already achieves a high F1-score before data augmentation, which limits the potential for further improvements.

Results by Imbalance Ratio and Sample Size

Figure 3.8 presents the minority and majority class F1-score for the "tor" base dataset, considering various combinations of sample size and imbalance ratio. Each line connects the results of a specific sample size with the three imbalance ratios tested. We can observe a pattern where, as the number of majority class examples increases and the dataset becomes

Table 3.10: Results categorized by dataset and majority class count.

dataset	maj_count	min_f1	min_f1_bal	min_f1_imp	maj_f1	maj_f1_bal	maj_f1_imp
cic-syscalls-adware	100	0.000000	0.222968	∞	0.897391	0.736370	0.82
	200	0.000000	0.240513	∞	0.897794	0.724752	0.81
	500	0.003419	0.235989	69.03	0.898319	0.837309	0.93
	1000	0.060914	0.330292	5.42	0.896864	0.828231	0.92
	1500	0.085570	0.308354	3.60	0.895584	0.859087	0.96
cic-syscallsbinders-adware	100	0.000000	0.278299	∞	0.897391	0.737301	0.82
	200	0.000000	0.186628	∞	0.897794	0.575843	0.64
	500	0.035659	0.320428	8.99	0.895777	0.810675	0.90
	1000	0.037592	0.343123	9.13	0.900721	0.855098	0.95
cic-syscallsbinders-smsmalware	100	0.000000	0.400140	∞	0.897391	0.677977	0.76
	200	0.000000	0.555040	∞	0.897794	0.845875	0.94
	500	0.287072	0.584644	2.04	0.928896	0.912317	0.98
	1000	0.491932	0.617447	1.26	0.936950	0.932369	1.00
cira	100	0.100000	0.591448	5.91	0.902484	0.873802	0.97
	200	0.279368	0.731822	2.62	0.929264	0.935837	1.01
	500	0.442151	0.867815	1.96	0.948800	0.972635	1.03
	1000	0.835905	0.931979	1.11	0.982123	0.987253	1.01
	2000	0.932897	0.970213	1.04	0.992618	0.995340	1.00
	4000	0.986268	0.986187	1.00	0.997877	0.997872	1.00
	8000	0.994862	0.994565	1.00	0.999127	0.998962	1.00
	16000	0.996410	0.997151	1.00	0.999386	0.999489	1.00
iscx-defacement	100	0.688980	0.684695	0.99	0.941888	0.922855	0.98
	200	0.675310	0.738958	1.09	0.954219	0.953249	1.00
	500	0.827021	0.803607	0.97	0.970576	0.965528	0.99
	1000	0.885730	0.891820	1.01	0.978409	0.980004	1.00
	2500	0.928648	0.920152	0.99	0.986058	0.985827	1.00
iscx-spam	100	0.259858	0.583519	2.25	0.930458	0.879538	0.95
	200	0.370092	0.689467	1.86	0.940072	0.943308	1.00
	500	0.602445	0.837459	1.39	0.958348	0.972204	1.01
	1000	0.872187	0.889902	1.02	0.979374	0.978951	1.00
	2500	0.921511	0.916000	0.99	0.985179	0.983167	1.00
tor	100	0.121831	0.597724	4.91	0.860685	0.848732	0.99
	200	0.097969	0.659305	6.73	0.903113	0.910294	1.01
	500	0.317785	0.703864	2.21	0.927671	0.927268	1.00
	1000	0.528781	0.705859	1.33	0.938525	0.925698	0.99
	2000	0.641171	0.729503	1.14	0.943386	0.938385	0.99
	4000	0.690154	0.728698	1.06	0.945129	0.942144	1.00
	8000	0.744699	0.734022	0.99	0.948662	0.943089	0.99

Table 3.11: Results by Dataset and by Imbalance Ratio.

dataset	imbalance_ratio	min_f1	min_f1_bal	min_f1_imp	maj_f1	maj_f1_bal	maj_f1_imp
cic-syscalls-adware	0.1	0.000000	0.095220	∞	0.952117	0.842450	0.88
	0.2	0.009853	0.250240	25.40	0.908223	0.840509	0.93
	0.4	0.080089	0.457411	5.71	0.831230	0.708490	0.85
cic-syscallsbinders-adware	0.1	0.000000	0.160598	∞	0.952069	0.820316	0.86
	0.2	0.000000	0.263419	∞	0.908475	0.740496	0.82
	0.4	0.054938	0.422342	7.69	0.833218	0.673376	0.81
cic-syscallsbinders-smsmalware	0.1	0.043510	0.401146	9.22	0.952570	0.821643	0.86
	0.2	0.137200	0.492670	3.59	0.915225	0.854405	0.93
	0.4	0.403543	0.724139	1.79	0.877977	0.850356	0.97
cira	0.1	0.574823	0.832116	1.45	0.978023	0.973777	1.00
	0.2	0.687224	0.895589	1.30	0.967017	0.968450	1.00
	0.4	0.825901	0.923987	1.12	0.961840	0.968218	1.01
iscx-defacement	0.1	0.736503	0.770211	1.05	0.981176	0.979633	1.00
	0.2	0.812423	0.790077	0.97	0.968482	0.958771	0.99
	0.4	0.854487	0.863251	1.01	0.949031	0.946074	1.00
iscx-spam	0.1	0.388165	0.675298	1.74	0.966832	0.948368	0.98
	0.2	0.559233	0.771611	1.38	0.949749	0.950434	1.00
	0.4	0.868258	0.902899	1.04	0.959478	0.955499	1.00
tor	0.1	0.263072	0.554668	2.11	0.947465	0.922879	0.97
	0.2	0.440626	0.708718	1.61	0.923714	0.918964	0.99
	0.4	0.643041	0.819032	1.27	0.900465	0.916275	1.02

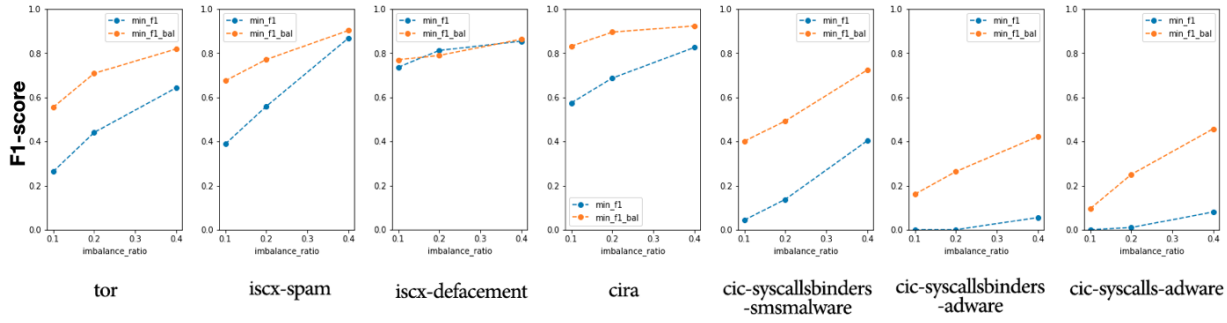

Figure 3.7: The F1-score for the minority class, before and after CGAN-based data augmentation, is shown for each dataset with varying imbalance ratios, averaged by majority class counts.

less imbalanced, the classifier's performance improves on the original imbalanced data. As the F1-scores on the imbalanced dataset become higher, the gains achieved through oversampling diminish because there is less room for improvement. This trend is consistent across all other datasets as well. It is important to note that the "iscx-defacement" datasets start with a relatively high F1-score, even under challenging conditions of high imbalance and small sample size. As a result, no improvements are observed when applying CGAN-based oversampling across different imbalance ratios and majority class counts in these datasets. For the majority class, the blue and orange lines exhibit similar patterns, suggesting that the F1-score remains relatively unchanged when CGAN-based oversampling is employed.

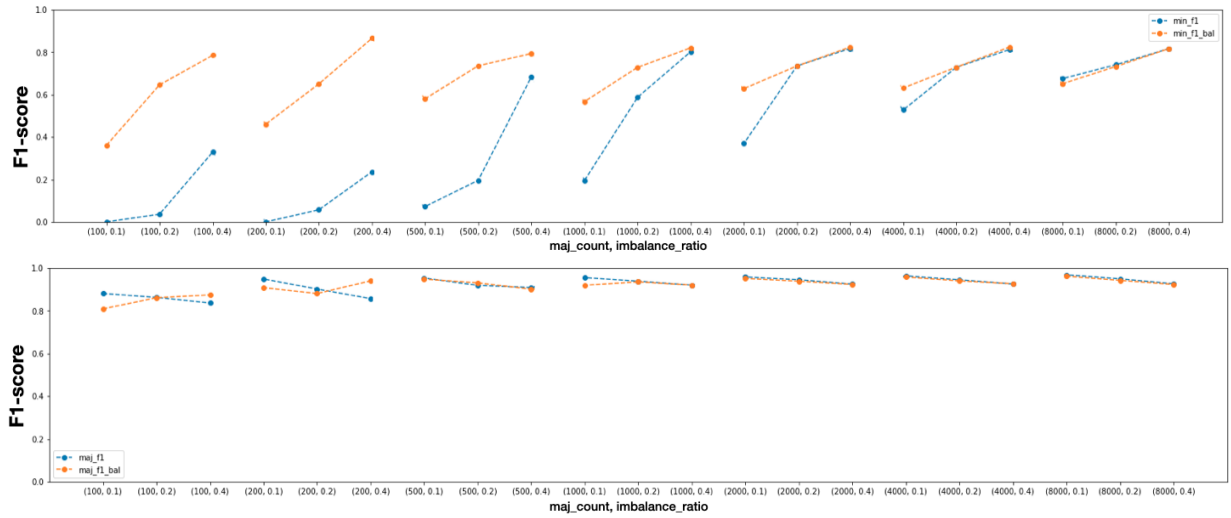

Figure 3.8: F1-score of the minority class on the "tor" dataset, before and after data augmentation, is shown with respect to the minority class count (top) and majority class count (bottom) and imbalance ratio.

Effect of Sample Size on Minority Class Performance

To investigate the trend seen with increasing sample sizes, we expanded our tests on the "tor" dataset by considering larger majority class counts, extending up to 50,000 samples. Specifically, we included previous experimental sample sizes of 100, 200, 500, 1,000, 2,000, and 4,000, and added results for 8,000, 16,000, 32,000, and 50,000 samples. The results are depicted in Figure 3.9. It can be observed that beyond a certain point, the classifier becomes proficient in distinguishing between the two classes in the imbalanced dataset.

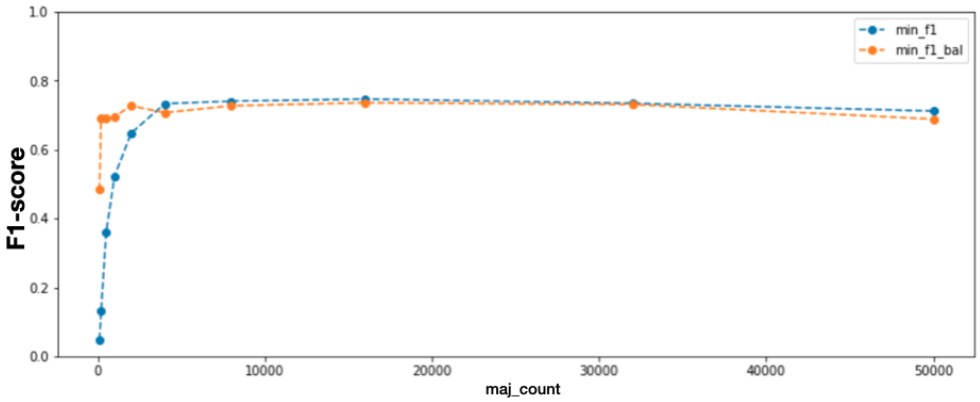

Figure 3.9: Expanded testing on increased majority class counts for the "tor" dataset

Consequently, the room for improvement diminishes, and the oversampling method does not enhance (nor degrade) the F1-score of the minority class.

3.6 Conclusion

In this chapter, we studied the suitability of CGAN as an oversampling strategy when the predictive task includes multiple data difficulty factors. This study aimed to shed light on the class imbalance problem in both imagery datasets and Cybersecurity tabular datasets. For imagery datasets, we examined the impact of four different data difficulty factors, including class overlap, data dimensionality, imbalance ratio, and sample size. Through experiments and analysis, we investigate how these factors influence the performance of classification models. In the case of Cybersecurity datasets, we specifically focused on two data difficulty factors: class imbalance and small sample size. By conducting experiments and evaluating the performance of different approaches, we aimed to gain insights into the challenges posed by these factors and explore effective strategies to address them.

In our imagery tests, we created 144 new datasets with specific attributes such as class overlap, data dimensionality, imbalance ratio, and sample size. Through extensive experimentation, we discovered that employing CGAN-based oversampling is an effective approach for addressing the studied data difficulty factors, resulting in significant improvements in the performance of the minority class across all scenarios. Overall, we found that this strategy performed well across various conditions. Specifically, CGAN demonstrated greater gains in challenging scenarios characterized by lower dimensionality, higher imbalance, higher class overlap, and smaller sample size. Remarkably, the most remarkable gains

were observed when using low-dimensional datasets. In conclusion, our experiments highlight the effectiveness of CGAN in addressing multiple data difficulty factors commonly encountered in real-world problems.

In our tests on the Cybersecurity domain, we constructed a data repository consisting of real-world Cybersecurity (tabular) problems to assess the effectiveness and reliability of the CGAN-based approach under different constraints. By generating 114 new datasets with controlled characteristics pertaining to imbalance ratio and small sample size, we conducted a comprehensive series of experiments. The results demonstrate the consistent enhancement of the minority class's performance through CGAN-based oversampling, particularly in challenging scenarios characterized by a higher imbalance ratio and smaller sample size. Furthermore, our findings indicate that although this approach does not yield further improvements in minority class performance as the sample size increases and the classes become more balanced, it also does not have a negative impact on the performance of the majority class.

3.7 Future Works

In future work, one path we might take is to investigate the compatibility of different GAN architectures and to conduct further experiments using both real-world and synthetic data sets. We can also contemplate an intriguing future research direction, namely the smart fusion of GANs with other oversampling methods.

One challenge we encountered in this chapter pertained to our setting, specifically with tabular non-imagery datasets (in our case the Cybersecurity datasets). Unlike imagery datasets where the visual assessment of the CGAN generator's outputs is possible, in the tested Cybersecurity datasets, this approach is not feasible. In imagery datasets, one can observe the generated images during the training process to assess the gradual improvement in realism and diversity. However, due to the nature of the Cybersecurity data in hand, this visual assessment is not applicable. As a result, in our experiments, we resorted to a fixed training duration of 1500 iterations through empirical trials. This challenge serves as the foundation for the subsequent chapter of the thesis. In Chapter 4, we will introduce a method that effectively addresses this issue by offering an efficient and convenient approach to training a GAN on both imagery and tabular data.

Chapter 4

Enhancing Generative Modeling: AutoGAN - An Automated Human-out-of-the-Loop Approach for Training Generative Adversarial Networks

GANs are complex to train, requiring careful monitoring to determine optimal training. Specifically, for an imagery dataset, a conventional approach to evaluate GAN outputs involves manual visual assessments to confirm their similarity to the original dataset. This often means allowing the GAN to produce several samples during each training iteration unit and, if the outcomes are not satisfactory, continuing the training. This research offers a solution that removes the need for this manual visual verification during training. Moreover, this method streamlines training GANs with tabular datasets. Recognizing the challenge of deciding when a GAN is adequately trained, in this chapter we introduce AutoGAN, an algorithm that automates this process with minimal human oversight. The work in this chapter is based on the following paper:

- Ehsan Nazari, Paula Branco, and Guy-Vincent Jourdan. Autogan: An automated human-out-of-the-loop approach for training generative adversarial networks. *Mathematics*, 11(4), 2023

4.1 Introduction

Generative models are essential components of numerous ML and computer vision algorithms, demonstrating remarkable success in diverse applications [9]. Among these models, GAN [38] have captivated the attention of the scientific community. GAN has been widely employed in real-world scenarios for data generation, including tasks such as image generation [13, 59–62], image-to-image translation [28, 55, 156], image super-resolution [71], video generation [88, 134], music generation [25], graph generation [8], and text generation [44]. Moreover, GANs has been successfully applied to address complex tasks such as data de-identification [99], oversampling [90, 91], enhancing classification accuracy [17, 102], handling missing values [81], trajectory prediction [45], and spatio-temporal prediction [114].

Training a GAN poses a greater level of complexity and challenge compared to training a standard ML algorithm [21, 84]. Unlike optimizing an objective function, GAN training involves a learning task that can be explained using game theory. This learning task is characterized as a minimax problem where two players engage in competition, with one player aiming to maximize an objective function while the other player endeavors to minimize it. The potential solution to this minimax problem is known as the Nash equilibrium of the game [35]. However, discovering the Nash equilibrium represents a complex task when compared to the optimization of an objective function [35, 38].

In addition to the complexity involved, training a GAN posses several other challenges. One of these challenges relates to the absence of a systematic criterion for determining when a GAN has reached a sufficient level of training for a given task during the training process. This is relevant when dealing with both imagery and tabular datasets. Different attempts have been made to tackle this issue, resulting in a growing body of research proposing multiple alternative measures. However, no single measure has emerged as the universally accepted method applicable to diverse applications and data modalities. Consequently, researchers and end-users are confronted with the task of determining when to halt GAN training through a non-systematic trial-and-error approach. This chapter aims to address this issue by seeking a systematic solution for determining the optimal point to stop GAN training, applicable across different data modalities.

The metrics currently used to address the aforementioned problem can be classified into two categories: qualitative evaluation and quantitative evaluation. Qualitative evaluation involves human judgment, while quantitative evaluation relies on mathematically defined distance functions. One commonly employed qualitative method to determine when to stop training a GAN is through visual inspection of the generated samples, which provides a direct and intuitive assessment of image quality [9]. However, this approach is expensive,

time-consuming, and not applicable to tabular data. Efforts have been made to regulate and standardize the visual inspection process in a systematic manner [9,10,109,155]. However, this solution faces several drawbacks that hinder its widespread adoption. These drawbacks include the difficulty of determining realism in certain domains like the medical field, limitations on the number of images that can be reviewed within a reasonable time frame, the potential incorporation of reviewer biases and opinions into the process [10], and the inability to apply visual inspection methods to tabular data.

In terms of quantitative evaluation, several measures have been proposed to assess a GAN trained on image datasets. Notably, IS [108] and FID [52] are popular measures in this category. These measures utilize a pre-trained model, such as InceptionNet, to derive a metric that evaluates the quality of generated images. Alternative solutions, like the FCD (explained in 4.2.2), replace the InceptionNet model with an Auto-Encoder. However, all these solutions produce a score, and it is ultimately the responsibility of the end-user to determine whether the GAN's training process can be concluded.

Furthermore, there are quantitative evaluation metrics that consider both imagery and non-imagery data. For example, classification accuracy-based measures are introduced in [55,106,122]. While quantitative measures do not encounter the same challenges as direct human (qualitative) evaluation, they may not directly align with human perceptions and judgments of generated samples [9].

Arjovsky et al. [3] proposed a new variant of GAN known as Wasserstein GAN (WGAN) to address the issue of oscillatory behavior observed in the loss values of GAN components. WGAN can be seen as an approach that incorporates qualitative measures into the loss functions of a GAN. The utilization of WGAN learning curves offers several benefits, such as enhancing the interpretability of sample quality. This, in turn, aids in addressing the challenge of determining the appropriate stopping point for GAN training. However, WGAN has a few limitations. It does not allow for direct comparison of results between different GAN architectures, the estimation of the Wasserstein distance may be imprecise [3], and for imagery datasets, WGAN still relies on human visual validation of the loss values.

In general, the proposed qualitative measures rely on human involvement during GAN training, while quantitative measurements that address some limitations of qualitative measures still necessitate human monitoring of the metric throughout training. Consequently, humans remain in the loop for both types of measures. Furthermore, qualitative measurements are exclusively applicable to GANs trained on imagery datasets, and the majority of popular quantitative measures are also designed for such datasets. As a result, the application of GANs to non-image datasets is restricted, and human inspection or supervision remains necessary. In this chapter, we bridge these gaps by introducing an al-

gorithm called AutoGAN, which operates without human intervention and fully automates the use of quantitative measures. In AutoGAN, the training of a GAN begins, and at each iteration, the improvements are evaluated using an oracle. The oracle, based on end-user preferences, assigns scores to generators based on their ability to synthesize "better" samples. AutoGAN relies on these oracle outputs to assess GAN training, even allowing for temporary performance deterioration while waiting for the GAN to recover from known oscillatory behavior. When no further improvements are observed for a certain number of consecutive iterations, AutoGAN returns the overall best GAN model obtained during the process. AutoGAN minimizes the need for human intervention and is applicable to various data modalities, including tabular and image data. Extensive experiments provide compelling evidence for the superiority of AutoGAN over GANs trained with meticulous human visual inspection of the generated images.

This chapter makes four contributions:

- It offers a literature review on two main topics: (i) various distances and performance measures for evaluating GAN performance, and (ii) existing algorithms that automate GAN processes.
- It introduces the AutoGAN Algorithm, a novel approach that automates the determination of when to stop GAN training, eliminating the need for human involvement. This algorithm can be applied to both imagery and tabular data modalities.
- It conducts extensive experiments using multiple imagery and tabular datasets, comparing various GAN evaluation metrics.
- It provides the complete code implementation of AutoGAN, ensuring its easy applicability and enabling the replication of the research findings.

The structure of this chapter is as follows. Section 4.2 provides an overview of the background and related works. In Section 4.3, we introduce our algorithm, AutoGAN, which addresses the problem of determining the optimal stopping point for GAN training automatically. Additionally, we present a collection of oracle instances that can be utilized within the AutoGAN Algorithm. The experiments conducted, including the datasets used and the experimental settings, are outlined in Section 4.4. The main results of the experiments are presented and discussed in Section 4.5. Finally, in Section 4.6, we conclude the chapter and in Section 4.7, we suggest potential avenues for future research.

4.2 Background and Related Work

In this section, we begin an examination of different distance measures and their application in evaluating GAN quality followed by a comprehensive explanation of how these measures can be employed. Additionally, we discuss relevant prior studies that explore architecture automation in GANs.

4.2.1 Quantifying GAN Quality: Important Metrics and Evaluation Criteria

In this subsection, we delve into various measures employed for assessing the quality and performance of GANs. Specifically, we examine three prominent measures: the Kullback-Leibler (KL) divergence, the Wasserstein distance, and the F1-score. The inclusion of the F1-score serves as an illustrative example of a performance assessment metric that holds potential significance in GAN-related tasks, particularly in imbalanced domains. While there exist other metrics, the F1-score stands out as one of the commonly utilized measures in GAN evaluations.

KL divergence is a widely recognized statistical measure that quantifies the similarity between two given distributions [68].

Let's consider two distributions, denoted as P and Q, with probability densities p and q respectively. The KL divergence is mathematically defined by Equation 4.1.

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \tag{4.1}$$

The Wasserstein distance offers an alternative approach to measure the dissimilarity between two probability distributions [105]. Consider two random variables, X and Y, following distributions P and Q respectively, with finite p-moments. Let $J(P, Q)$ denote the set of all joint distributions J for the random variables (X, Y). The p-Wasserstein distance is defined by Equation (4.2).

$$W_p(P, Q) = \left(\inf_{J \in J(P,Q)} \int \|x - y\|^p dJ(x, y) \right)^{1/p} \tag{4.2}$$

where $p \geq 1$. Specifically, when $p = 1$, it corresponds to the Earth Mover or 1-Wasserstein distance. Similarly, the 2-Wasserstein distance is commonly referred to as the Fr'echet

distance.

To evaluate the performance of a model where the focus of the end-user is on a minority class, the F1-score is a useful metric. It relies on the concepts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). TP represents correctly classified positive cases, TN represents correctly classified negative cases, FP represents cases incorrectly classified as positive, and FN represents cases incorrectly classified as negative. The F1-score (cf. Equation 4.5) is calculated as the harmonic mean of precision (cf. Equation 4.3) and recall (cf. Equation 4.4). The F1-score can also be calculated for each class in the domain, and variants such as macro-average or micro-average can be used to obtain a global F1-score for multi-class problems.

$$\text{precision} = \frac{TP}{TP+FP} \tag{4.3}$$

$$\text{recall} = \frac{TP}{TP+FN} \tag{4.4}$$

$$\text{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.5}$$

4.2.2 Experimental Configurations for Evaluating GAN Performance

Various distances and measures can be employed to evaluate the performance of a GAN. However, the specific configurations in which these distances or measures are applied can vary significantly. This section examines these configurations, considering the experimental settings they promote and the underlying assumptions they rely on. We will explore six different configurations in detail, explaining how they are utilized to assess the performance of a GAN.

Classification Accuracy Score: Training on Synthetic Data and Testing on Real Data

Ravuri and Vinyals [106] propose that if a generative model effectively captures the data distribution, downstream tasks should perform similarly regardless of whether they use generated or original data. They train class-conditional generative models, including CGANs and Variational Auto-Encoders, on labeled real data. Subsequently, a classifier is trained on synthetic data and applied to predict the labels of real images. The resulting model

can be evaluated using performance metrics such as the F1-score. This configuration, referred to as CAS-real, has also been explored in [122], where it highlights the diversity of samples generated by a CGAN. Although CAS-real can be applied to a CGAN trained on various data types, it necessitates a labeled multi-class dataset. Additionally, it is limited to CGAN (i.e. this method cannot be used for a vanilla GAN), which may be a drawback in certain deployment scenarios.

Classification Accuracy Score: Training on Real Data and Testing on Synthetic Data

If we assume that the generated images from a well-trained CGAN are realistic, then classifiers trained on real images should also be able to correctly identify the synthesized images. This assumption, proposed by [55], forms the basis of the CAS-syn configuration. In CAS-syn, the roles of synthetic and real data are switched compared to the CAS-real configuration. In CAS-syn, a classifier is trained on real data and tested on synthetic data, whereas CAS-real uses real data for training and synthetic data for testing. The CAS-syn configuration has also been explored in [122], where the authors argue that it can assess how closely the generated samples approximate the unknown real distribution in image data. CAS-syn can be applied to a CGAN trained on both imagery and tabular data. However, it requires a multi-class labeled dataset and is applicable only to CGAN (i.e. this method is not applicable to a vanilla GAN).

Inception Score

Salimans et al. [108] introduced an alternative approach called the Inception Score (IS). This method involves computing the conditional label distribution, $p(y, x)$, by applying an InceptionNet model to each image generated by a GAN trained on the original dataset. The underlying assumptions of this approach are twofold: (i) The GAN should generate images with high confidence for each label, implying that $p(y|x)$ should have low entropy; (ii) The GAN should produce diverse images, indicated by a high entropy in the marginal distribution $\int p(x = G(z)), dz$. Based on these requirements, the authors propose a metric in the form of $\exp[\mathbb{E}x \sim p_g DKL(p(y|x)||p(y))]$. They argue that this metric exhibits a direct correlation with human judgment. To facilitate comparison, the expected value of the KL-divergence is exponentiated.

Despite being a widely used approach, it is important to consider several weaknesses associated with this method. These disadvantages, as outlined in [10], include the following:

(1) Sensitivity to model parameters and their implementations; (2) Bias towards the ImageNet dataset and InceptionNet models; (3) Inability to capture diversity within classes; (4) Requirement of a large sample size for reliable results; (5) Potential for achieving a high Inception Score by inputting only one example per ImageNet class; (6) Limitation to GANs trained on specific imagery datasets, as the InceptionNet model relies on colored images as inputs, employs Convolutional layers, and is trained on the ImageNet dataset. Consequently, the Inception Score cannot be applied to black-and-white images or tabular data. Additionally, using the Inception Score with a GAN trained on colored images other than ImageNet can lead to misleading interpretations [6].

Confidence and Diversity Score

The proposed methodology for obtaining the CDS introduces a different configuration compared to IS. The CDS configuration involves the following steps: (1) Utilizing a neural network classifier of any architecture; and (2) Training the classifier on the specific dataset of interest, rather than relying on the ImageNet dataset [95]. This modification allows CDS to be applicable to GANs both trained on imagery and non-imagery data, whereas the IS is limited to a specific domain of imagery data. However, it is important to note that the CDS requires the data to be categorized into two or more classes and necessitates the availability of class labels to calculate the score. In contrast, the original IS does not rely on class labels from the target dataset.

Fréchet Inception Distance

Heusel et al. [52] propose an approach to evaluate the quality of images generated by a GAN by utilizing the extracted features from both real and generated data. Typically, an InceptionNet model is trained on the ImageNet dataset, and the features obtained from the last pooling layer prior to the output classification layer are considered for evaluation.

The extracted features from the real data (X_r) and the generated data (X_g) are treated as continuous multivariate Gaussian distributions. Specifically, X_r is assumed to follow a Gaussian distribution $\mathcal{N}(\mu_r, \Sigma_r)$, where μ_r represents the mean and Σ_r denotes the covariance matrix. Similarly, X_g is assumed to follow another Gaussian distribution $\mathcal{N}(\mu_g, \Sigma_g)$, with μ_g as the mean and Σ_g as the covariance matrix. The parameters of these underlying distributions are estimated, and the Fr'echet distance between the two distributions is computed. The FID is employed to measure the similarity between these multivariate Gaussian distributions, and its calculation for X_r and X_g is shown in Equation (4.6).

$$d^2 = |\mu_X - \mu_Y|^2 + tr(\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{1/2}) \tag{4.6}$$

FID provides a measure of the similarity between the estimated distributions, where a smaller FID indicates a closer match between the two distributions and better output from the GAN [10]. Unlike IS, FID metric considers the diversity within classes, but it is also subject to biases and can be overestimated with small sample sizes [10]. Both FID and IS rely on the InceptionNet model; IS uses the pre-trained model as is, and FID utilizes the model as a feature extractor. Since the InceptionNet is trained on colored images from the ImageNet dataset, IS and FID may be limited to colored images. However, FID has the potential for broader applicability as it treats InceptionNet solely as a feature extractor. To test this hypothesis, we will conduct experiments using black-and-white images to evaluate the versatility of FID.

Fréchet Confidence and Diversity Score

Obukhov and Krasnyanskiy proposed modifications to FID metric, resulting in FCD score [95]. Instead of using the InceptionNet model, an Auto-Encoder model trained on the target data is employed. The Auto-Encoder's encoder component serves as the feature extractor, and different sizes of the encoder's output layer, representing the number of extracted features, are explored. The authors observed that the FCD score correlates with the quality of generated images, as demonstrated in their work [95]. However, this correlation does not hold for generated tabular and non-imagery data. This conclusion is based on the absence of a correlation between the FCD score and the reduction in errors of the generator and discriminator. Nevertheless, it is important to note that the oscillatory behavior of the generator and discriminator should be taken into account, as mentioned in prior research [3]. Decreasing errors does not necessarily indicate proper training. In Section 4.5, we will delve into these findings further when presenting our experimental results.

4.2.3 Tabular Data Generation

The generation of tabular data serves various purposes, including dataset balancing for classification tasks or ensuring data privacy through de-identification. Methods like Copulas [115], which model the interdependence of multiple random variables, and GANs, can be used in this context. The Python package, Synthetic Data Vault [100], employs the aforementioned methods including a variation of GANs to approximate the distribution of tabular data. This package requires specifying the number of training iterations, with

a default set at 300. However, our approach, discussed in Section 4.3, introduces a more sophisticated technique for determining GAN training iterations, moving beyond fixed or hyperparameter-determined values.

4.2.4 Algorithmic Automation in GANs

Various attempts have been made to automate the search for optimal generator architectures in GANs. Neural architecture search methods have demonstrated promising results by surpassing manually designed architectures across multiple tasks [32, 133]. Wang et al. introduced an algorithm for automated neural architecture search in deep generative models [140]. Similarly, Gong et al. defined a search space for generator architectural variations and employed a Recurrent Neural Network (RNN) to guide the search process [34]. Their approach utilized the IS as a reward and adopted a multi-level search strategy to progressively explore neural architectures. However, there are still numerous research directions to explore in this domain, including expanding the search space, extending the search to the discriminator, incorporating class labels, and testing the search on high-resolution images.

Automation in GAN research brings about additional concerns, particularly regarding the size of the models. Expanding on the advancements in neural architecture search, Fu et al. introduced the AutoGAN-Distiller (AGD), the first AutoML framework specifically designed for GAN compression. However, several aspects still require further investigation. One significant challenge in training a GAN with a predefined architecture is determining the appropriate stopping point, indicating when the GAN has achieved optimal training. Surprisingly, no attempts have been made to automate this process for a given architecture. Currently, in many image-related applications, the evaluation of image quality and the decision to terminate the GAN training procedure heavily rely on domain experts. These experts must subjectively assess when the generated images meet the desired quality criteria, leading to potential biases and time-consuming analysis. Moreover, this inspection-based approach is not applicable when dealing with non-image data. To address these concerns, we present the AutoGAN solution in the following section, aiming to tackle these issues and provide automated answers.

4.3 Proposed AutoGAN Method

In this section, we present our proposed solution called AutoGAN, which aims to automate the training process of GANs. We start by outlining the objectives and criteria that

AutoGAN aims to fulfill, followed by a comprehensive explanation of our algorithm. Additionally, we provide a thorough demonstration of several oracle instances, which play a crucial role within the AutoGAN framework. It is important to note that the term "oracle" is used in its traditional sense from Computing Theory, representing a machine that can efficiently solve a decision problem in constant time (e.g., as described in [97]).

4.3.1 Objectives and Prerequisites of the Algorithm

Our solution aims to deliver a well-trained GAN model to the end-user that is specifically tailored for their target task. To achieve this, the end-user is required to provide the task requirements and specify the GAN architecture to be trained. Once these settings are provided, the AutoGAN algorithm takes over the process, and the end-user's intervention is no longer necessary, except for the initial design choices. The AutoGAN algorithm utilizes the provided information to generate a trained GAN model that aligns with the end-user's preferences. This fully automated approach eliminates the need for human intervention, such as determining the number of epochs or manually inspecting results at specific checkpoints to decide when to stop the training process.

AutoGAN incorporates a crucial element known as the oracle, which plays a pivotal role in the system. The oracle is specifically tailored by the end-user to encapsulate the task requirements they wish to address. It is the responsibility of the end-user to define an appropriate oracle that aligns with their desired task. In essence, the oracle functions as a mechanism that takes the GAN generator as input and produces a corresponding score indicative of the generator's ability to generate high-quality samples. Simply put, the oracle assesses the GAN's quality, a role that traditionally involves humans manually reviewing the GAN's output images, specifically when trained on image datasets. The introduction of the oracle streamlines GAN training by minimizing human intervention. The end-user has control over the oracle's parameters, allowing customization according to their needs. For a comprehensive understanding of the oracle's functionality within AutoGAN, please refer to Section 4.3.3, which provides an explanation of several oracle instances that can be utilized in diverse contexts and serve different objectives.

4.3.2 The AutoGAN Algorithm

In the case of training GANs on imagery datasets, a human is usually involved in the process of determining when to stop training by visually assessing the generated samples. However, this inspection is not feasible when working with tabular data, making the task

more challenging. While certain metrics can be monitored, human intervention is still required to decide when to halt the training process. It is common to see researchers imposing arbitrary limits on the number of training epochs, resulting in sub-optimal GAN models.

To tackle this issue, we present AutoGAN, a systematic solution that automates the training process of GANs using a quantitative measure. Our approach removes the reliance on human judgment to determine the appropriate number of training epochs and when to stop the GAN training. AutoGAN provides an automated and fully autonomous method for end-users to determine the optimal stopping point of GAN training, without the need for data inspection, image analysis, or metric evaluation.

The core idea behind the AutoGAN algorithm is to enable the continued training of a GAN, even when there is no apparent improvement in the generated samples after multiple iterations. The objective of AutoGAN is to afford the GAN sufficient opportunities to surpass any possible local sub-optimalities and ultimately achieve an enhanced model. The functionality of AutoGAN relies on the end-user's selection of an oracle instance. By utilizing the provided oracle instance, AutoGAN iteratively evaluates the quality of the output samples until further enhancement in the GAN's performance is not expected. At each iteration, the oracle produces a scalar score that represents the quantitative measure chosen by the end-user to assess the GAN's performance.

Algorithm 4.1 illustrates the pseudo-code of our proposed solution. We consider four inputs for the algorithm: (1) the maximum number of failed attempts, which determines the duration without observing any improvements in the GAN that we are willing to tolerate; (2) the unit used for the training iterations; (3) an untrained GAN with a specific architecture; and (4) an instance of an oracle that encompasses the task requirements specified by the end-user.

The training process of the GAN occurs iteratively after initialization. In each iteration, the GAN is trained and evaluated based on the oracle settings. If a superior solution is found, it is saved as the best-trained GAN up to that point, along with the corresponding highest achieved score. However, if the obtained GAN performs worse than the currently stored best GAN, it is discarded, and a new iteration begins. This local deterioration in performance is an expected occurrence due to the relationship between the losses of the generator and discriminator [21]. Thus, we anticipate a certain number of consecutive unsuccessful attempts. Nevertheless, once the maximum number of failed attempts is reached, indicating that the GAN has been trained for the specified maximum number of consecutive rounds without any improvements, the algorithm terminates and outputs the best model obtained during the process.

Algorithm 4.1 The AutoGAN algorithm

Input: $Max\_failed\_attempts$: The maximum number of failed attempts accepted;
$Train\_unit$: The unit used for training iterations;
$Untrained\_GAN$: The GAN architecture selected and not trained;
$Oracle$: The selected oracle instance;
Output: $best\_GAN$: The trained GAN model

```
current_GAN ← Untrained_GAN
best_GAN ← Untrained_GAN
best_score ← −∞
failed_attempts ← 0
while failed_attempts ≤ Max_failed_attempts do
    Train current_GAN for one Train_unit
    score ← oracle(current_GAN)
    if score ≥ best_score then
        failed_attempts ← 0 ;    // the new trained GAN is better than the best
         known
        best_GAN ← current_GAN
        best_score ← score
    else
        failed_attempts ← failed_attempts + 1 ; // the trained GAN is worst than
         the best known
    end
end
return best_GAN
```

4.3.3 Potential Oracle Instances

One vital aspect of our algorithm revolves around the selection of an appropriate oracle instance. While this responsibility falls on the end-user, it requires careful consideration, as the evaluation of AutoGAN relies heavily on the chosen oracle. In this section, we present different oracle instances that can be utilized in diverse contexts, taking into account specific data characteristics and GAN requirements. Specifically, we will outline oracle instances that make different assumptions regarding the data's attributes (such as modalities or the availability of class labels) and the GAN architectures involved.

As mentioned earlier, an oracle is a function that evaluates a given generator and assigns a score corresponding to the quality of the generated samples. The specific definition of

what constitutes a "better" sample is determined by the oracle instance chosen by the end-user. For example, in certain cases, a "better" sample may refer to a sample that closely approximates the distribution of the dataset. However, in other scenarios, a "better" sample might indicate increased diversity in the data or, in the case of images, higher quality and sharpness. Figures 4.1–4.5 offer an overview of the six oracle instances, which will be further discussed in subsequent sections to provide a more comprehensive understanding.

The choice of an oracle instance for a specific task is a reflection of the end-user's preferences and needs to be adaptable to various deployment scenarios. It is important to note that the oracle instance encompasses not only the metric or score used but also the entire architecture employed to calculate that score. For example, different oracle instances may involve the use of certain or all layers of a specific neural network, or employ a distinct classifier tailored for the task at hand. Therefore, the oracle entity encompasses multiple settings that can and should be adjusted to the particular problem being addressed, distinguishing the oracle instances from existing fixed solutions.

Oracle Instance Based on CAS-Real

This oracle instance initiates by utilizing the available labeled data to train a CGAN. Subsequently, the CGAN is prompted to generate a labeled dataset, which is then utilized to train a classifier. The classifier is trained using the generated data. Finally, the performance of the trained classifier is assessed on the real labeled dataset, employing a chosen performance evaluation metric. In our specific implementation, we have opted for the F1-score as the metric of choice. This selection was made as we intended to evaluate this oracle instance in the context of an imbalanced problem, where the F1-score proves to be a suitable measure. In this particular case, a higher CAS-real score indicates higher quality of the CGAN outputs. The structure of this oracle instance based on CAS-real is depicted in Figure 4.1.

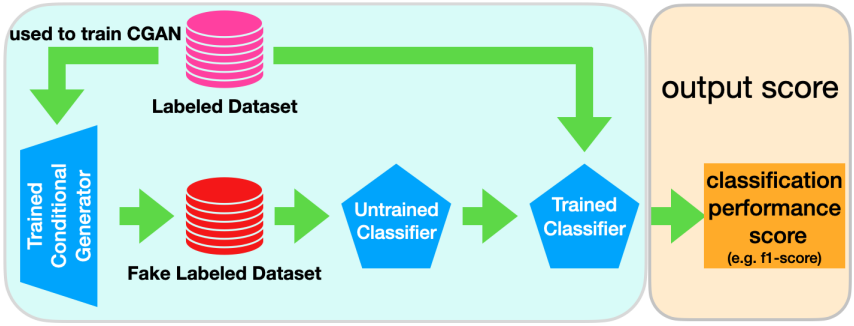

Figure 4.1: The architecture of an oracle instance based on CAS-real.

Oracle Instance Based on CAS-syn

This oracle instance operates similarly to the previous one based on CAS-real, with the main distinction being a reversal in the roles of real and synthetically generated data. The process begins by training a classifier using real labeled data. Simultaneously, the same real data is employed to train a CGAN. The generator component of the CGAN is then utilized to generate a synthetic test set, which is subsequently employed to evaluate the performance of the trained classifier. The classifier's performance on the generated labeled test set is measured using a chosen performance assessment metric. In our specific implementation, we have selected the F1-score due to its suitability for imbalanced domains. In this case, a higher CAS-syn score corresponds to higher quality outputs from the CGAN. Figure 4.2 provides an overview of an oracle instance based on CAS-syn.

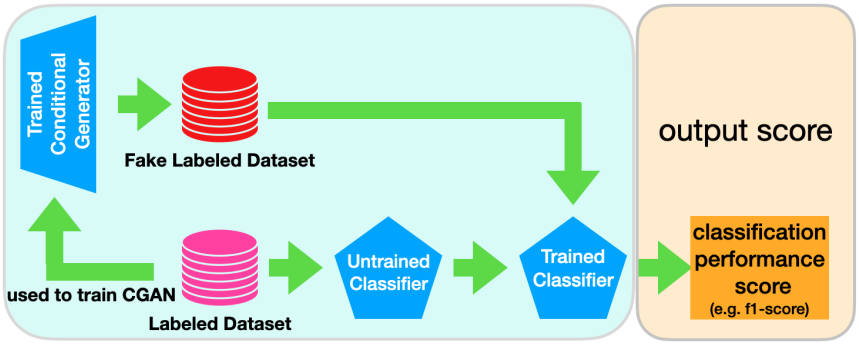

Figure 4.2: The architecture of an oracle instance based on CAS-syn.

Oracle Instance Based on Inception Score

In this oracle instance, the InceptionNet model is initially trained on the ImageNet dataset, while the GAN is trained on a provided real dataset. Subsequently, the generator component of the GAN is employed to generate new samples, which are then fed into the InceptionNet model. The IS is calculated based on the model's output vectors, serving as a measure to evaluate the quality of the GAN. A higher IS value indicates better quality outputs from the GAN. This particular oracle instance can be utilized with conditional and non-conditional GANs. Figure 4.3 presents an overview of the described oracle instance. However, it is important to note that this IS-based oracle is not suitable for tabular data or black-and-white images, as detailed in Section 4.2.2.

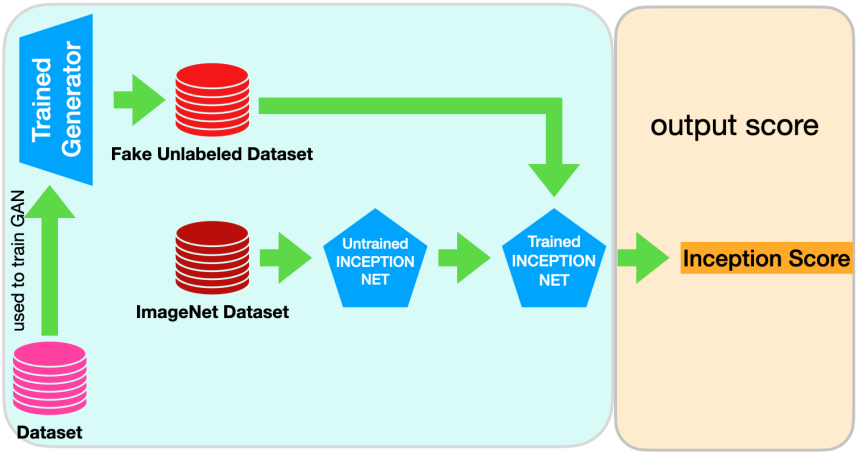

Figure 4.3: The architecture of the oracle based on IS.

Oracle Instance Based on Fréchet Inception Distance

This oracle instance also utilizes the InceptionNet model; however, it employs a truncated version of this neural network architecture. In this case, the oracle employs a pre-trained InceptionNet model on the ImageNet dataset. Subsequently, a GAN is trained on the target real dataset, and the generator component of the GAN is employed to generate a specific number of fake samples. Both the real and fake data are then processed by the truncated version of the InceptionNet model to obtain their respective InceptionNet-represented features. The features extracted from both the fake and real data are used to calculate the FID score. In this case, a lower FID score indicates higher quality outputs from the GAN. In our implementation, to establish a positive correlation between the FID

score and performance, we multiplied the FID score by minus one. This oracle instance can be used with both conditional and non-conditional GANs. Figure 4.4 provides an overview of the oracle instance based on FID.

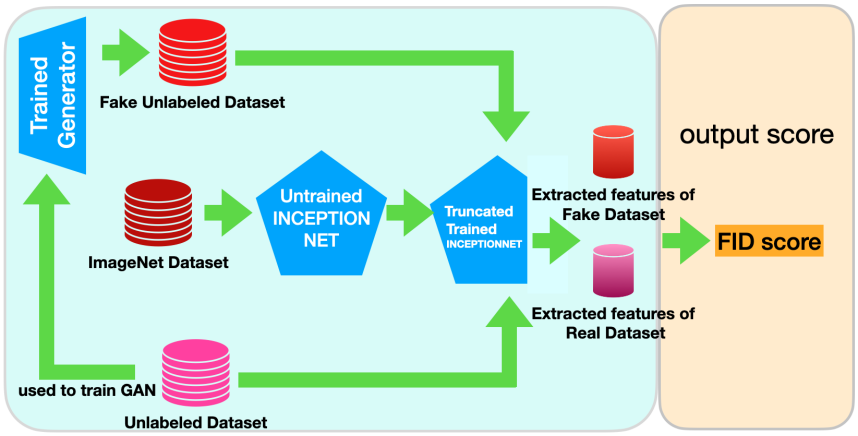

Figure 4.4: The architecture of an oracle instance based on FID.

Oracle Instance Based on Confidence and Diversity Score

Since its introduction in [108], IS has become widely utilized for evaluating generative models in the context of image data. Numerous studies have employed IS metric for assessing the performance of image generative models [43, 55, 59, 156]. The InceptionNet model, utilized in computing the IS, consists of 2D-Convolutional networks that are well-suited for processing images. It is important to note that the IS metric relies on an InceptionNet model pre-trained on the ImageNet dataset. Consequently, as mentioned in [95], the IS metric cannot be directly applied to real-valued tabular data.

In [95], the authors explore the applicability of the CDS metric on both imagery and tabular datasets by replacing the InceptionNet model with a neural network classifier trained specifically on the target dataset. Figure 4.5 illustrates an oracle instance we developed that utilizes the CDS metric for testing purposes on both imagery and tabular data. This oracle instance involves training a neural network classifier and a GAN model using the end-user dataset. The GAN's generator is then employed to generate a set of synthetic samples, which serves as the test set for the classifier. Subsequently, CDS is calculated based on the performance of the classifier on this test set. A higher CDS value indicates better quality of the GAN outputs. This oracle can be used with conditional and non-conditional GANs.

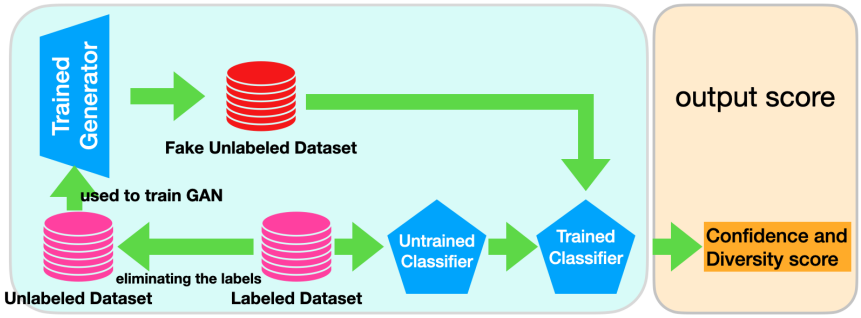

Figure 4.5: The architecture of an oracle instance based on CDS.

Oracle Instance Based on Fréchet Confidence and Diversity Score

In the research conducted by Morozov et al. [87], a comprehensive analysis is conducted to compare various metrics with the original FID. The authors examine multiple measurements that rely on the InceptionNet model pre-trained on different datasets. Additionally, they explore other metrics that utilize self-supervised models as feature extractors. Moreover, the applicability of FCD is investigated in [95], considering both imagery and tabular data.

This oracle instance is built upon FCD score and commences by training a GAN and an Auto-Encoder using the available real dataset. Subsequently, the generator of the GAN is employed to generate synthetic data. Both the real and generated data are then fed into the encoder, which extracts a new representation of their features. Finally, the extracted features from both datasets are utilized to compute the FCD score. A lower FCD indicates higher quality GAN outputs. Similar to the modified FID metric proposed in [95], we implemented an oracle instance based on FCD, as depicted in Figure 4.6. Additionally, as a minor implementation detail, we multiply FCD score by minus one, enabling us to observe a positive correlation between this score and output quality. This oracle instance is compatible with both conditional and non-conditional GANs.

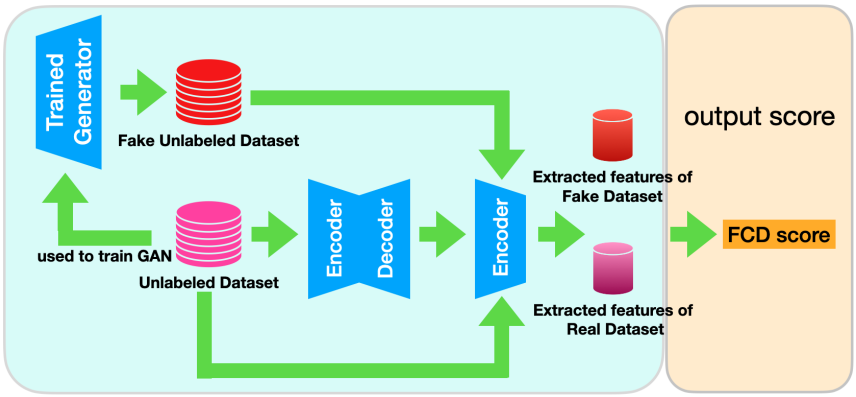

Figure 4.6: The architecture of an oracle instance based on FCD.

Summary of the Oracle Instances

The various oracle instances described have different assumptions regarding the data and GAN architecture, making them applicable in different contexts. These instances serve as examples of potential oracle instances that can be utilized. However, the selection of the appropriate oracle instance for a specific GAN training task should be driven by the task requirements and end-user preferences. Table 4.1 provides a comparison of the different oracle instances used in this paper, highlighting their characteristics. It can be observed that some oracles are exclusively designed for use with CGANs (CAS-real and CAS-syn), while others can be applied to both conditional and non-conditional GANs. Regarding the requirement of class labels, only FCD does not necessitate labeled data during training, whereas the other alternatives discussed rely on labeled data. Furthermore, all oracles can be applied to imagery data, while only four of them are applicable to tabular data. The table also indicates the number of times the network/classifier in each oracle needs to be trained (column "train times"). "One time" indicates that it is trained initially and requires no further training, while "multiple" implies that training must be repeated after each change in GAN weights. CAS-real is the only oracle instance that requires multiple training iterations to obtain the desired score, whereas the other oracles only need to be trained once. Comparing all six implemented oracle instances, it is evident that FCD is the most versatile, working for both tabular and image data, not requiring class labels, and being compatible with any GAN architecture. It is worth noting that all oracle instances can be used with colored images, and most of them can also be used with black-and-white images. However, preliminary tests indicate that the IS oracle is not suitable for black-and-white images.

Table 4.1: A comparison between the requirements of the described oracle instances.

Oracle Instance	Required Labels		Type of GAN	Type of Data			Train Times	Metric	Source
	Labeled Data during Training	Labeled Data to Generate Score		Imagery Color	Imagery B & W	Tabular			
CAS-real	Required	Required	Requires a CGAN	✓	✓	✓	Multiple	F1-score	[106, 122]
CAS-syn	Required	Not required	Requires a CGAN	✓	✓	✓	One time	F1-score	[55, 122]
IS	Required	Not required	Any GAN	✓			One time	KL-divergence	[108]
FID	Required	Not required	Any GAN	✓	✓		One time	Fréchet distance	[52]
CDS	Required	Not required	Any GAN	✓	✓	✓	One time	KL-divergence	[95]
FCD	Not required	Not required	Any GAN	✓	✓	✓	One time	Fréchet distance	[95]

4.4 Experimental Evaluation

This section details the series of experiments conducted to evaluate the effectiveness of the AutoGAN algorithm proposed in this study. We start by presenting an overview of our experiments, followed by a description of the utilized datasets and an explanation of the experimental configurations. To ensure the reproducibility of our findings, we have made all the codes employed in this research openly accessible to the research community. The code can be found at https://github.com/enazari/autoGAN.

4.4.1 Experiments' Overview

Our objective is to evaluate the effectiveness of the AutoGAN algorithm (see Algorithm 4.1) in addressing the problem of determining when to stop training a GAN. In order to determine if a GAN is adequately trained or if the training process should be automatically halted at an optimal point, we utilize the GAN as an oversampling tool in the presence of class imbalance. Our hypothesis is that a well-trained GAN will yield similar benefits to those achieved by GANs that have been meticulously trained with human intervention. By employing this experimental setup, we can assess the efficacy of our proposed solution in both imagery and tabular datasets.

The experimental design can be summarized as follows. Initially, we evaluate the performance of a specific classifier on an imbalanced domain to establish a baseline performance (referred to as the Initial method). Subsequently, we train various GAN models using different stopping criteria to determine when the training process should be terminated. These GAN models are then utilized to generate synthetic samples, which are employed to balance the number of instances in the training set. We assess the performance of classifiers

trained on the different balanced training sets to determine the quality of the synthetic data generated by the GANs and ascertain whether they were effectively trained.

It is important to emphasize that the synthetic instances generated by the GAN are solely incorporated into the training set, while the test set remains unchanged and separate from the rest of the data, exclusively used for evaluating the classifier. Our main focus is to examine whether AutoGAN can identify an appropriate stopping point that results in a well-trained GAN. If the performance outcomes of our proposed method are at least comparable to those of the alternative methods, we can confidently assert that we have effectively resolved the challenge of determining when to stop training a GAN.

In our experimental setup, we employ different alternative methods to determine the stopping point for GAN training. The specific alternatives considered depend on the type of dataset utilized, whether it is imagery or tabular data. In total, we evaluate four primary alternatives, which are as follows:

1. **Initial**: this serves as a baseline approach where the original imbalanced dataset is used to train a classifier;
2. **Fixed**: the GAN is trained for a fixed number of iterations, which is determined based on previous research papers or experimental guidelines that have demonstrated successful GAN results;
3. **Manual**: the GAN is trained with human visual inspection, where experts visually assess the training process to determine the number of iterations required for achieving a well-trained GAN;
4. **AutoGAN**: the GAN is automatically trained using the AutoGAN algorithm in combination with one of the defined oracle instances described in Section 4.3.3.

We compare the performance achieved by these different methods against each other and the baseline approach (Initial) to evaluate their effectiveness.

To ensure fair comparisons, we initiate the training process of a specific GAN and simultaneously employ the different alternative methods to determine the appropriate stopping point, as illustrated in Figure 4.7. This allows us to obtain the final state of the GAN using various methods while following the same training process. The training process of the GAN is only concluded when all the tested methods indicate a stopping signal. Through this framework, we can evaluate the quality of GANs trained under different processes on both imagery and tabular datasets. Additionally, we can observe the number of epochs utilized by the different methods and their impact on performance.

Figure 4.7: An instance illustrating when each alternative method could indicate the need to stop training for a specific GAN.

We conducted three main experiments, summarized in Table 4.2. The first experiment focused on tabular datasets and involved the initial, fixed, and AutoGAN methods. Due to the nature of tabular data, manual inspection is not feasible as it is not possible to visually assess the quality of generated tabular data. For the fixed method, we determined the fixed number of stopping iterations based on previous experiments in [91]. In the AutoGAN method, we could only apply four of the Oracles described in Section 4.3.3, namely CAS-real, CAS-syn, CDS, and FCD, which are suitable for tabular datasets.

In the second and third experiments, we used imagery datasets. The second experiment involved extensive testing on a large number of imagery datasets, where we applied the initial, fixed, and AutoGAN methods with six different oracles. For the third experiment, which was conducted on a smaller subset of the imagery datasets, we tested all the methods used in the second experiment, including the manual method. However, the manual method could not be executed for all the imagery datasets due to the significant time required. Manual inspection involves a human evaluating the quality of images generated by the GAN after each iteration. To manage the time constraints, we decided to run the manual method for a subset of the imagery datasets. This allowed us to include a commonly used method for assessing GAN quality in our tests while keeping the experiment duration manageable.

Table 4.2: An overview of the three main experiments, including both tabular and imagery datasets.

Experiment	Experiment #1	Experiment #2	Experiment #3
Data Used	Tabular Data	Imagery Data 1	Imagery Data 2
Alternative Methods	Initial Fixed AutoGAN-CAS-real AutoGAN-CAS-syn AutoGAN-CDS AutoGAN-FCD	Initial Fixed AutoGAN-CAS-real AutoGAN-CAS-syn AutoGAN-CDS AutoGAN-FCD AutoGAN-FID AutoGAN-IS *	Initial Fixed Manual AutoGAN-CAS-real AutoGAN-CAS-syn AutoGAN-CDS AutoGAN-FCD AutoGAN-FID

* AutoGAN-IS was used with colored imagery datasets only.

4.4.2 Datasets

We examined a total of eight base datasets in our experiments, consisting of four tabular datasets and four imagery datasets. We created binary versions for each of these base datasets, resulting in a combined total of 17 binary datasets. Figure 4.8 provides an overview of these datasets, illustrating the eight base datasets and their corresponding binary versions. The orange branches represent the four base tabular datasets, which serve as the foundation for constructing multiple binary datasets displayed in the red branches. Similarly, the blue branches indicate the four base imagery datasets, with their respective binary versions depicted in the purple branches. The names of the datasets and the two classes chosen for the binary task are indicated in the red and purple branches. In total, our analysis encompasses 17 binary datasets, comprising seven tabular datasets and ten imagery datasets.

We create a class imbalance scenario where the two classes of the predictive tasks are not equally represented in the available data. This means that one class has a higher number of examples (referred to as the majority or negative class), while the other class is poorly represented (known as the minority or positive class). To generate class imbalance tasks with different characteristics, we produced multiple variations by manipulating the number of majority and minority class examples.

For each of the 17 binary datasets, we generated 16 variants by modifying both the imbalance ratio and the sample size. The imbalance ratio is defined as the ratio between the number of minority class samples and the number of majority class samples. To obtain these dataset variants, we followed the procedure described in [91]. Specifically, we first selected a random sample of majority class examples that matched a desired count for the majority class. Then, we randomly selected a set of minority class examples from the dataset and combined them with the previously selected majority class examples to achieve the desired imbalance ratio.

To cover a range of scenarios, we considered all combinations of four different majority class counts (100, 200, 500, and 1000) and four imbalance ratios (0.1, 0.2, 0.3, and 0.4). This resulted in a total of 272 imbalanced datasets (17 binary datasets $\times$ 16 variants) that were generated and utilized in our experiments.

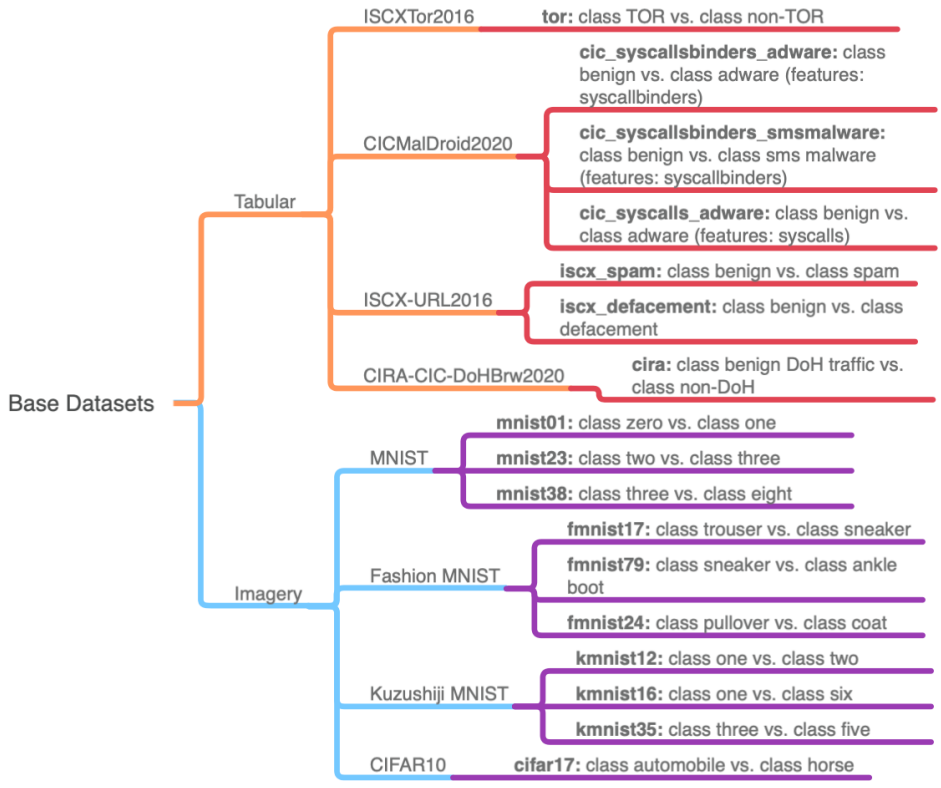

Figure 4.8: The blue and orange branches symbolize the base datasets for creating 17 binary datasets, shown in red and purple branches, with dataset names and classes in the rightmost branch.

In the upcoming sections, we will delve into a comprehensive explanation of the base tabular and imagery datasets, as well as the 17 binary datasets that have been derived from them and are employed in our experiments.

4.4.2.1 Tabular Datasets

We employed the same set of four base tabular datasets as utilized in [91]: ISCXTor2016 [46], CICMalDroid2020 [82], ISCX-URL2016 [83], and CIRA-CIC-DoHBrw2020 [86]. From these base datasets, we created a total of seven distinct binary datasets.

The dataset ISCXTor2016 [46] comprises features extracted from network traffic using the ISCXFlowMeter tool. From this dataset, we created a binary classification dataset named "tor" by assigning labels to the target variable for TOR and non-TOR traffic.

The dataset CICMalDroid2020 [82] contains information about Android malware and consists of five classes: benign, SMS malware, riskware, adware, and banking. The dataset encompasses features extracted from 11,598 APK files and is divided into two categories: (i) syscallbinders, which includes the frequencies of system calls, binders, and composite behavior, with a total of 470 features, and (ii) syscalls, which includes the frequencies of system calls, with a total of 139 features.

Using the features from the syscallbinders category, we created two binary datasets: cic-syscallsbinders-adware and cic-syscallsbinders-smsadware. cic-syscallsbinders-adware dataset consists of instances with classes benign and adware, while cic-syscallsbinders-smsadware dataset includes instances with classes benign and SMS malware.

Additionally, we generated another binary dataset using the features from the syscalls category, named cic-syscalls-adware, which includes instances from the adware and benign classes.

From the base dataset ISCX-URL2016 [83], which originally consisted of five different classes, we derived two binary datasets: iscx-spam and iscx-defacement. The iscx-spam dataset includes instances from the benign and spam classes, while the iscx-defacement dataset comprises instances from the benign and defacement classes.

The final base tabular dataset we took into account is the CIRA-CIC-DoHBrw2020 [86], which features a target class with three labels: benign DoH traffic, malicious DoH traffic, and non-DoH traffic. From this dataset, we constructed a binary dataset called "cira" that exclusively encompasses instances from the benign DoH and non-DoH classes.

4.4.2.2 Imagery Datasets

During our experiments, we incorporated four base imagery datasets: MNIST [24], Fashion-MNIST [145], Kuzushiji-MNIST [20], and CIFAR10 [67].

MNIST is a dataset comprising 70,000 black-and-white images of handwritten digits, encompassing 10 different classes. Each image has dimensions of 28×28 pixels. From the MNIST dataset, we derived three binary datasets with distinct class combinations. The first dataset, mnist01, consists of instances with classes zero and one. These classes were chosen as they are considered the simplest binary choice among the ten classes. The second dataset, mnist23, includes instances with digits two and three. These classes were selected as they have more complex shapes compared to the simpler digits in the previous case. Lastly, the mnist38 dataset comprises instances from class three and class eight. By selecting different classes for these three binary datasets, we aim to achieve varying levels of complexity in the predictive tasks. We posit that distinguishing between classes 0 and 1 is the simplest task while distinguishing between classes 3 and 8 is considered the most challenging task.

Fashion-MNIST dataset consists of 70,000 black-and-white images representing 10 distinct clothing classes, each image being 28×28 pixels in size. Similar to the approach taken with the MNIST dataset, we created three binary datasets from Fashion-MNIST by selecting different classes, aiming to achieve tasks of varying complexity. The following datasets were generated: (1) fmnist17, which includes instances from the classes trousers and sneakers; (2) fmnist79, comprising instances from the classes sneaker and ankle boot; and (3) fmnist24, consisting of instances from the classes pullover and coat. Based on our hypothesis, the fmnist17 task is expected to be the easiest among the three datasets, fmnist79 is anticipated to have an intermediate level of complexity, and fmnist24 is projected to be the most challenging task.

The Kuzushiji-MNIST dataset comprises 70,000 black-and-white images featuring 10 classes of handwritten Hiragana Japanese syllabary. Each image in the dataset has dimensions of 28×28 pixels. Similarly to the previous datasets, we generated three distinct datasets by selecting two classes from the original dataset, resulting in the following datasets: (1) kmnist12, which includes instances from classes 1 and 2; (2) kmnist16, consisting of examples from classes 1 and 6; and (3) kmnist35, comprising cases from classes 3 and 5. For each case, we randomly chose the two classes used in the respective dataset.

Lastly, we created a binary dataset named cifar17 from the CIFAR10 dataset. CIFAR10 consists of 60,000 colored images representing 10 different classes of objects, with each image having dimensions of $32 \times 32 \times 3$ pixels. From this dataset, we extracted the automobile and horse classes to construct the cifar17 binary dataset.

4.4.3 Experimental Setting

In our study, as illustrated in Figure 4.9, we conducted three experiments involving a total of 17 base datasets. Each dataset has 16 variants with varying levels of class imbalance and sample sizes (refer to Section 4.4.2 for more details). In each experiment, we examined different alternative methods to determine when the GAN should stop generating images. The first method called "initial", involved using the original imbalanced training set. The remaining methods utilized images generated by the GAN, halted with a specific stopping signal, to oversample the training set and create a balanced distribution between the two classes. To evaluate these methods, we employed different oracles in conjunction with the AutoGAN algorithm. Furthermore, we explored both fixed stopping criteria (based on a predetermined number of iterations) and manual inspection by human observers as additional approaches.

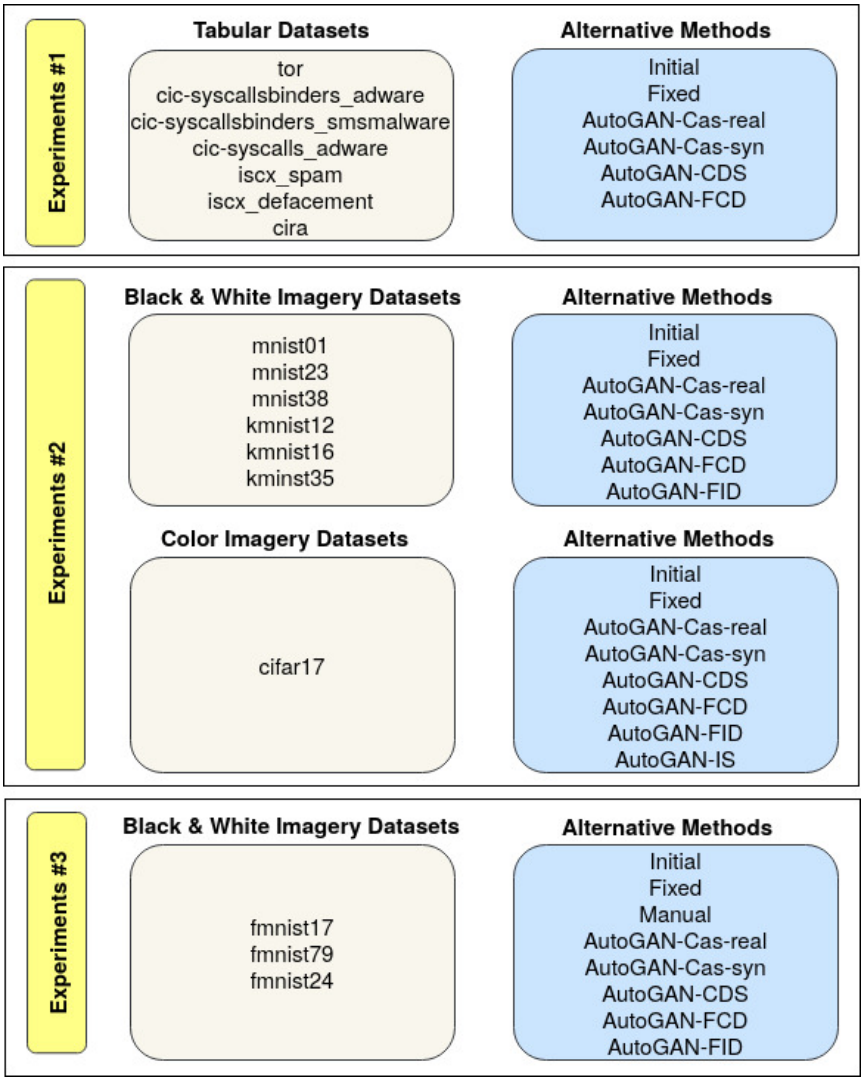

Figure 4.9: Detailed overview of the three primary experiments conducted.

A single GAN was trained until all methods indicated a stopping signal, as previously explained and depicted in Figure 4.7. To ensure robust evaluation, we employed a stratified five-fold cross-validation procedure. The F1-scores for both the minority and majority classes were recorded, regardless of the internal metric used by the AutoGAN algorithm.

The results presented include the average and standard deviation of the five-fold outcomes. Additionally, we provide information about the number of training iterations re-

quired for the GAN under each alternative method. This allows us to examine the relationship between performance and the duration of training.

Furthermore, we conducted an analysis of the Pearson correlation between the results obtained from the different alternative methods for stopping the GAN training that were tested. This analysis provides insights into the consistency and agreement among the various methods.

For our experiments, we opted for a CGAN variation of a GAN. We explored two primary architectures: (i) fully connected hidden layers for both the discriminator and the generator, and (ii) Convolutional layers for both the generator and discriminator. The output layer of the generator and the input layer of the discriminator were adjusted to align with the specific number of features in each dataset. Complete architectural specifications can be found in Appendix A.

The details of the parameters utilized for AutoGAN, including the implemented oracles, can be found in Appendix A.1.1. In the case of the fixed alternative, we employed the parameters previously utilized in [91] for the tabular datasets. However, for the imagery datasets, we determined these values through an experimental approach involving trial and error. Regarding the manual alternative, we relied on inputs provided by a human expert.

For the classification task, we opted for a fully connected deep neural network. The configuration of the network, including the number of hidden layers and perceptrons in each layer, was customized to suit the specific requirements of each dataset. A comprehensive description of the network architecture can be found in Appendix A.2.

4.5 Results and Discussion

In this section, we present a summary of the key findings and conclusions drawn from the three conducted experiments.

4.5.1 Experiment #1: Tabular Datasets

The results from the tabular datasets revealed three distinct patterns, and we will illustrate these patterns by focusing on the results of the tor, iscx_defacement, and cira-based datasets as representative examples.

4.5.1.1 tor-based datasets

A similar pattern observed in the tor-based datasets can also be seen in the cic-syscallsbinders-adware, cic-syscallsbinders-smsmalware, and cic-syscalls-adware datasets. The average F1-scores for the minority class, obtained by training the neural network on various variants of the tor dataset, are depicted in Figure 4.10. The figure illustrates that all alternative methods employed for GAN training yielded superior results compared to the initial setting without oversampling. This indicates that all methods successfully trained the GAN to generate high-quality images. The results are further grouped by majority class count in Figure4.11 and by imbalance ratio in Figure 4.12.

The results provide the following insights: (1) When using CGAN as an oversampling technique, there is minimal to no deterioration in the F1-scores of the majority class, while the F1-scores of the minority class show significant improvement. (2) Increasing difficulty factors such as higher imbalance ratios and smaller sample sizes lead to more challenging classification tasks, resulting in lower F1-scores. (3) Lower initial F1-scores correspond to greater improvement through CGAN oversampling. (4) The AutoGAN algorithm shows comparable results to the fixed setting in terms of F1-scores. (5) AutoGAN algorithm achieves these results with fewer iterations for training the GAN when using three out of the four tested oracles (FCD, CAS-real, and CDS).

The plots on the right side of Figures 4.11 and 4.12 demonstrate distinct stopping iterations for different oracle instances. For instance, even though the F1-scores of CAS-real and CAS-syn oracles are comparable for both the minority and majority classes, CAS-real stops at approximately 1,000 iterations, while CAS-syn stops at around 2,500 iterations. Moreover, CAS-real achieves similar F1-scores as the fixed method but requires fewer training iterations.

Figure 4.10: The box plot illustrates the F1-scores of the minority class obtained from different stopping methods applied to tor-based datasets.

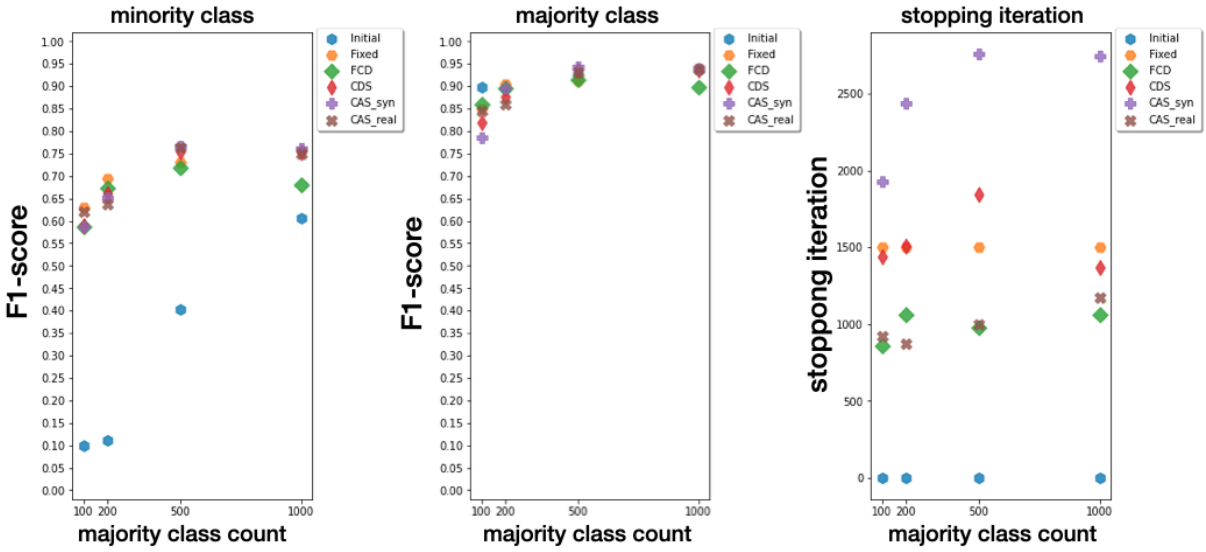

Figure 4.11: F1-score results for tor-based datasets are presented, with minority class scores on the left, majority class in the center (based on majority class count), and average iterations to stop training on the right.

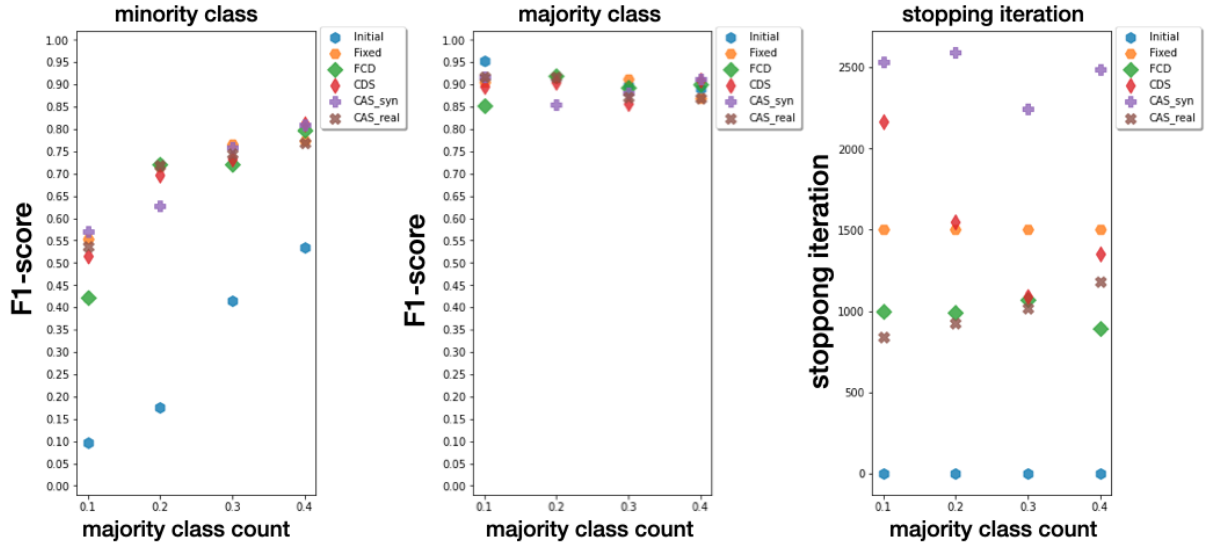

Figure 4.12: F1-score results for tor-based datasets are presented, with minority class scores on the left, majority class in the center (based on imbalance ratio), and average iterations to stop training on the right.

4.5.1.2 iscx_defacement-based datasets

Once the F1-score reaches a certain threshold, the potential for further improvement diminishes. This means that beyond that point, there are fewer opportunities for enhancement. The experiments conducted with iscx_defacement-based datasets, as summarized in Figures 4.13 and 4.14, indicate that none of the trained GANs show improvement compared to the initial setting. The F1-scores obtained from the AutoGAN algorithm are also similar to those of the fixed setting, although there are some variations in the number of iterations required for GAN training. While the fixed method necessitates 1,500 iterations, other methods such as CDS consistently require fewer iterations.

4.5.1.3 cira-based datasets

The same pattern observed in cira-based datasets was also found in iscx-spam-based datasets. The three observations identified in tor-based datasets can also be observed here (Figures 4.15 and 4.16). Additionally, the results indicate that the AutoGAN algorithm consistently outperforms the fixed technique in terms of performance. Furthermore, when compared to the fixed method, the results obtained by AutoGAN with FCD and CDS oracles are superior while requiring fewer iterations. In fact, if we consider AutoGAN with FCD oracle, it achieves one of the best F1-scores despite only requiring around 500 training iterations to stop the GAN (compared to the fixed method's 1,500 iterations).

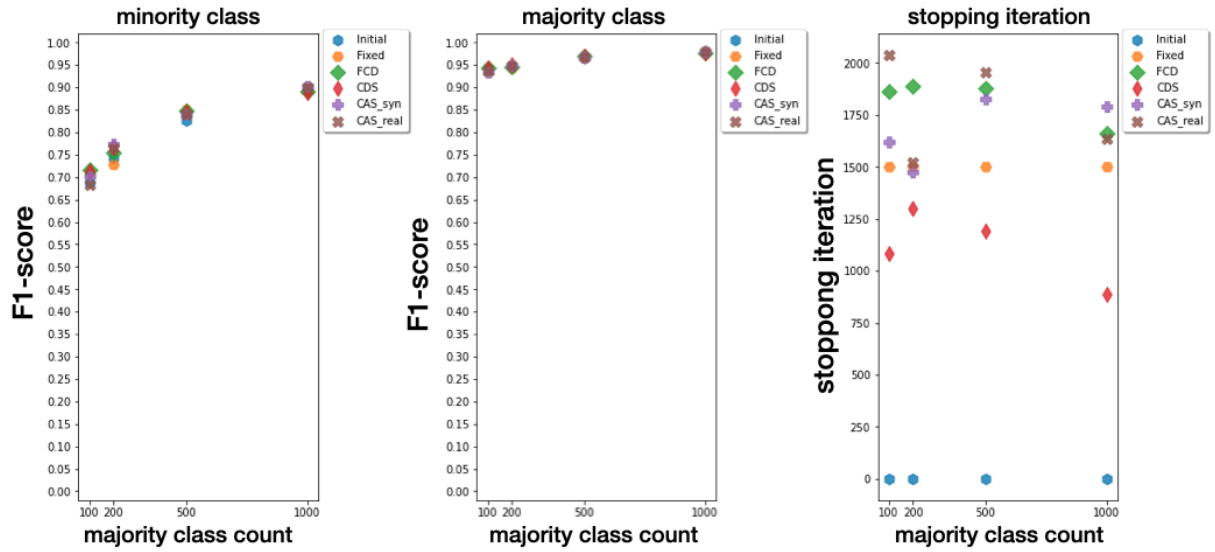

Figure 4.13: F1-score results for iscx_defacement-based datasets by majority class count (minority on the left, majority on the center) and average stopping iterations (right).

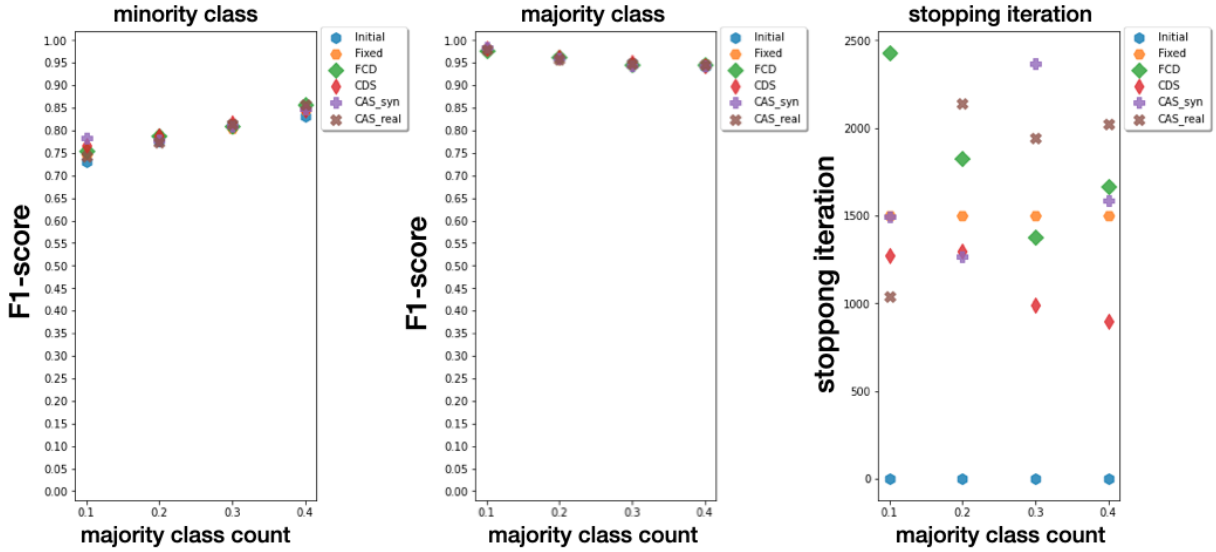

Figure 4.14: F1-score results for iscx_defacement-based datasets by imbalance ratio (minority on the left, majority on the center) and average stopping iterations (right).

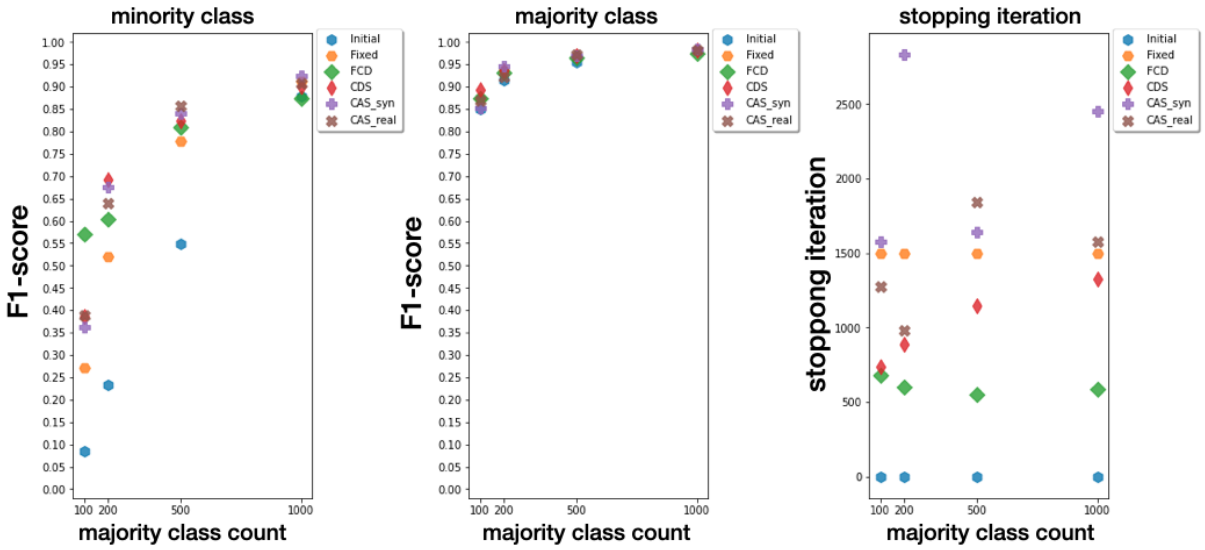

Figure 4.15: F1-score results for cira-based datasets by majority class count (minority on the left, majority on the center) and average stopping iterations (right).

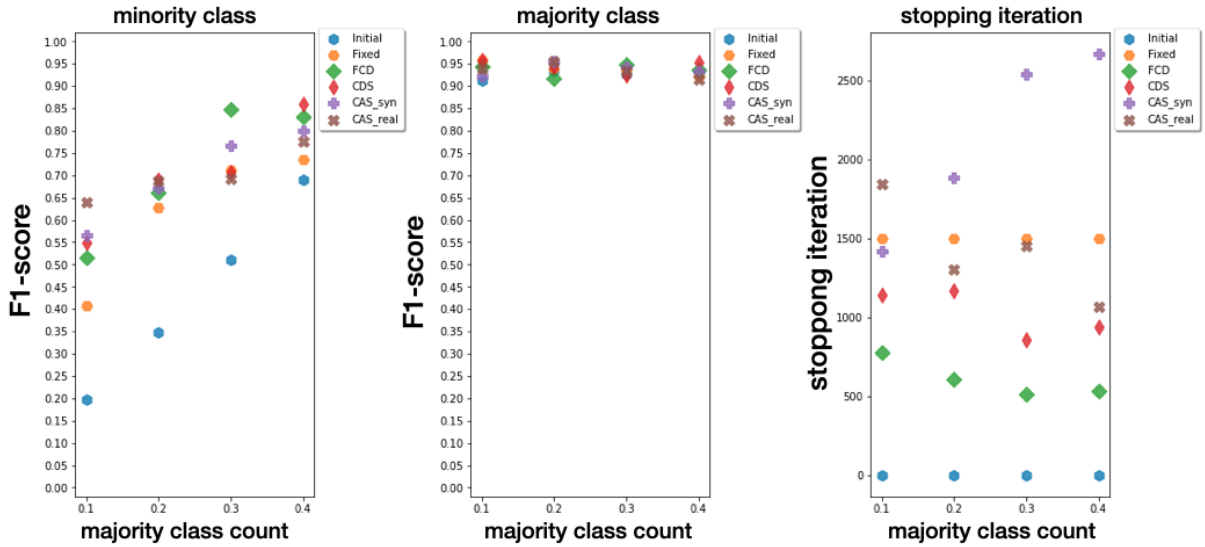

Figure 4.16: F1-score results for cira-based datasets by imbalance ratio (minority on the left, majority on the center) and average stopping iterations (right).

4.5.2 Experiment #2: Imagery Datasets

4.5.2.1 Black-and-White Imagery Datasets

kmnist-based datasets. The kmnist-based datasets were selected to present the experiment results, while similar results were obtained with the mnist-based datasets. In Figure 4.17, the results of all kmnist12 variants are shown by imbalance ratio. It is evident that all alternative methods outperform the initial approach. Furthermore, AutoGAN algorithm with different oracles achieves comparable performance to the fixed alternative for both minority and majority classes. However, the advantage of using AutoGAN lies in the significantly lower number of training iterations required. While the fixed approach used over 7,000 iterations, the automatic method used between 1,000 and 4,000 iterations with the FCD and FID oracles, and other oracles fell within this range.

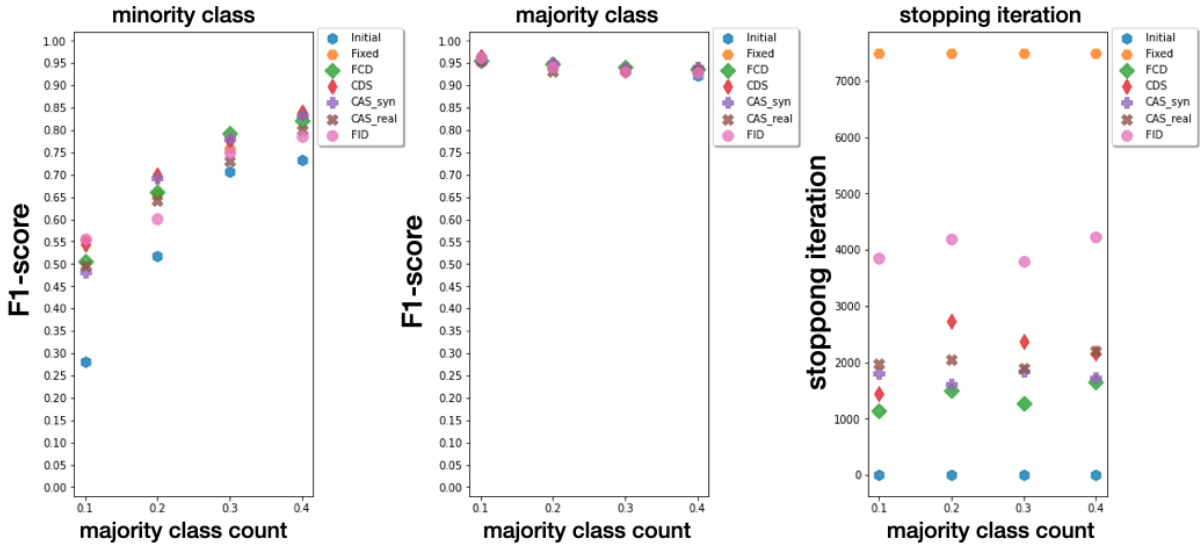

Figure 4.17: F1-score results for kmnist12-based datasets by imbalance ratio (minority on the left, majority on the center) and average stopping iterations (right).

4.5.2.2 Colored Imagery Datasets

The results of the experiments conducted on the cifar17 dataset's 16 variants are presented in Figure 4.18. Overall, we do not observe significant improvements in the oversampling technique. However, in the case of extreme class count (e.g., 100 for the majority class),

we notice improvements in the minority class with minimal impact on the majority class. In this scenario, the AutoGAN approach achieves comparable performance to the fixed iteration method while requiring fewer training iterations on average (refer to Figure 4.18 on the right). It is worth noting that, unlike black-and-white imagery datasets, the IS oracle can also be applied to this dataset. Additionally, the results demonstrate that all tested oracles require less training to achieve similar results. Specifically, FID and FCD utilize between 5,000 to 10,000 iterations, while the fixed method demands 20,000 iterations. This highlights the adaptability of our AutoGAN, which can dynamically adjust the required training iterations without human intervention. Notably, the CDS and IS oracles require a higher number of training iterations for cases with 100 and 200 majority class counts, respectively, which were determined automatically.

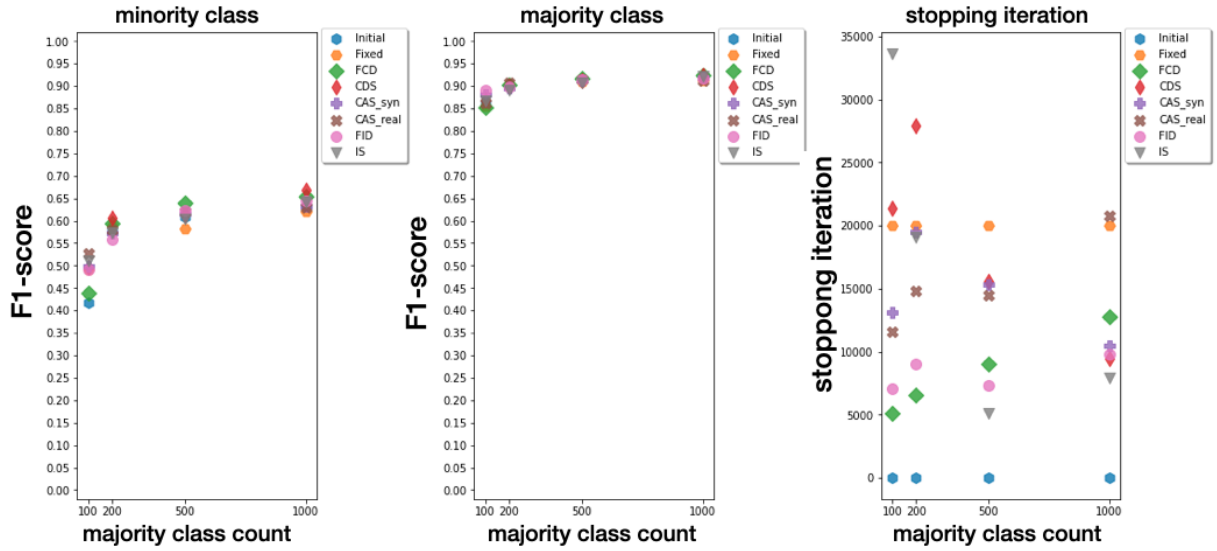

Figure 4.18: F1-score results for cifar17-based datasets by imbalance ratio (minority on the left, majority on the center) and average stopping iterations (right).

4.5.3 Experiment #3: Imagery Datasets with Human Inspection

fmnist dataset was utilized to create three binary class problems: fmnist17, fmnist79, and fmnist24. The selection of classes aimed to present varying levels of difficulty for human visual distinction between the two classes. fmnist17 dataset exhibits a clear visual distinction between the classes, while fmnist79 dataset poses a more challenging task. fmnist24 dataset represents the most difficult task in terms of visual classification. Figure 4.19

showcases the overall F1-score results and training iterations required for these datasets. The experiments reveal that the classification difficulty for a neural network classifier is equivalent to that of human visual classification. This observation is evident in the leftmost plots of Figure 4.19, where the easiest task yields high performance across all methods, the intermediate task shows some performance degradation, and the most difficult task demonstrates significantly lower F1-scores for all methods. These findings clearly indicate that as the classification task becomes more challenging, the F1-score of the minority class deteriorates. GAN over-sampling provides the most notable benefits in the most difficult case, where there is considerable room for improvement.

The following key observations can be derived from these findings: (1) the performance of the majority class remains largely unchanged in almost all cases. (2) The results of AutoGAN algorithm closely resemble those obtained through human manual inspection and the fixed number of iterations. (3) A comparison between manual inspection and fixed iterations reveals similar performance results with fewer iterations for the manual approach. (4) AutoGAN algorithm, with any of the tested oracles, achieves equivalent performance to the manual method with a lower number of training iterations. This is a remarkable outcome, as AutoGAN can halt training to attain the same performance as human inspection of generated images, but with less training.

4.5.4 Correlation Analysis of the Stopping Methods Used

We conducted a thorough analysis of the correlation between different methods used to stop GAN training. Specifically, we focused on the F1-scores of the minority class and compared the results of AutoGAN with various oracles to other methods such as initial, fixed, and manual. To assess the correlation, we calculated the Pearson correlation coefficient for four groups of experimental results. These groups were based on the tabular datasets from experiment #1, black-and-white imagery datasets from experiment #2 and fmnist-based datasets, colored imagery datasets from experiment #2, and fmnist-based datasets tested with the manual option in experiment #3. The correlation results and corresponding dendrogram for the four groups can be seen in Figures 4.20–4.23, respectively.

Figures 4.20, 4.21, and 4.23 reveal that the initial method shows the lowest correlation with the other methods. However, an exception to this trend is observed in group 3, which consists of colored imagery datasets, as depicted in Figure 4.22. We propose that this difference can be attributed to the results of over-sampling (Figure 4.18). In cases where the initial results are poor and there is significant room for improvement, GAN-oversampling demonstrates notable enhancements. Conversely, with cifar17-based datasets (group 3),

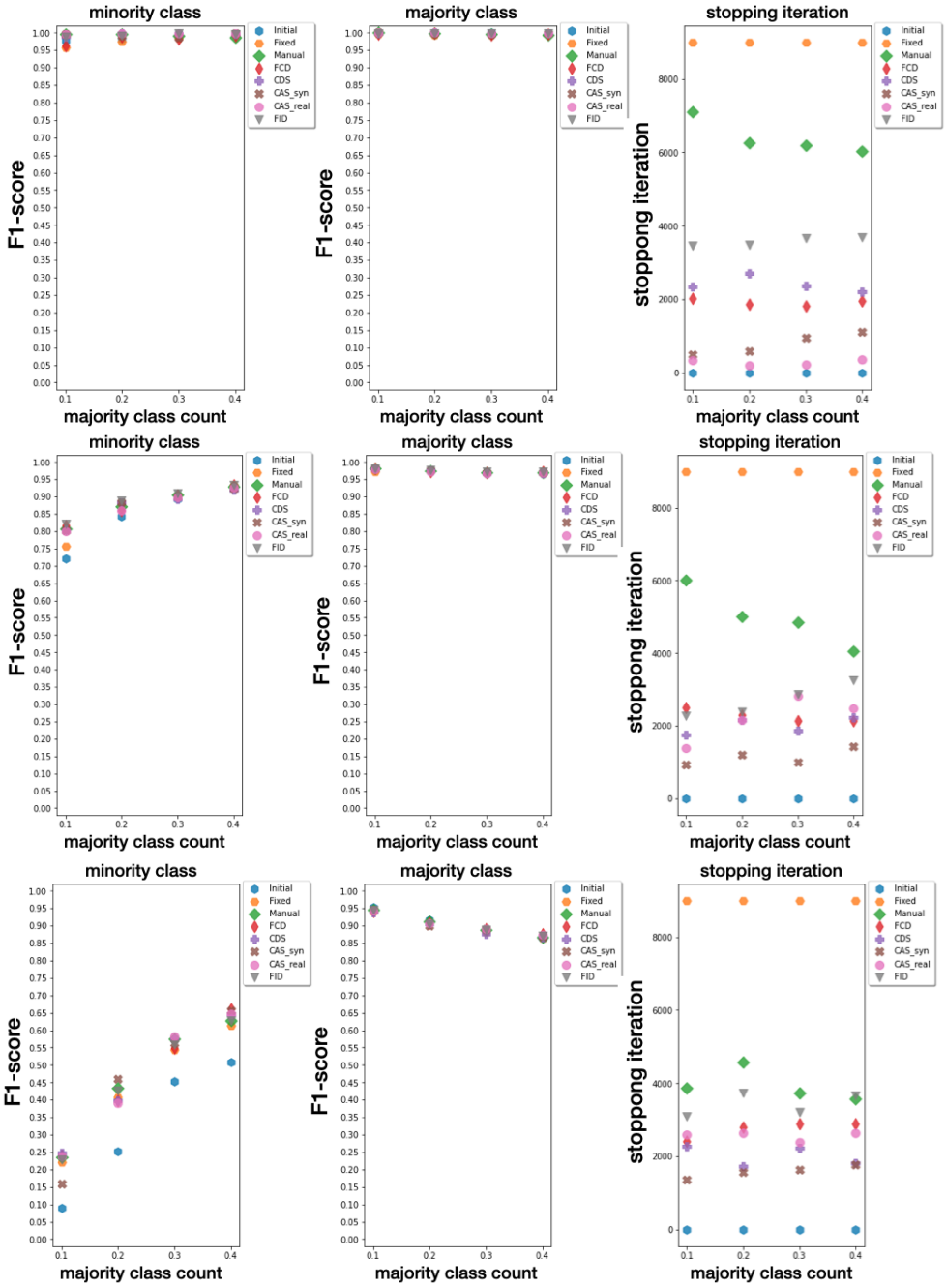

Figure 4.19: Average F1-scores for fmnist17, fmnist79, and fmnist24 datasets, categorized by imbalance ratio, are shown: minority class (left), majority class (center), and average iterations to halt training (right), for top, middle, and bottom datasets respectively.

GAN-oversampling yields minimal or no improvement. This lack of improvement from the initial results suggests a higher correlation between the initial approach and the alternative methods in this scenario.

Figure 4.20 clearly indicates that all the methods are distinctly more separated from each other compared to the initial method. In contrast, the figure also emphasizes that the fixed method shows a higher level of correlation with the other methods, particularly with the cluster that includes CDS and CAS-syn.

Figure 4.21 demonstrates the greatest distinction between the initial method and the other methods. Additionally, it is noteworthy that the fixed method exhibits a closer correlation with FCD and FID for these experiments.

The correlation results of group 3, as shown in Figure 4.22, are particularly surprising. Not only is the initial method not clearly separated from the other methods, but there is also a strong correlation observed between IS and fixed. It should be noted that IS was only used with one colored imagery dataset (cifar17), so these results may be influenced by the specific performance of these datasets.

Lastly, the correlation results presented in Figure 4.23 confirm the close relationship between the fixed method and FID and FCD oracles. Furthermore, these results highlight the strong correlation between the manual method and CAS-syn, as well as CDS and CAS-real. This reinforces the advantages of replacing manual inspection of images with our AutoGAN algorithm, which can achieve highly correlated results with several of the implemented and tested oracles. Moreover, this is accomplished with a reduced number of training iterations, resulting in benefits in terms of computational time and memory.

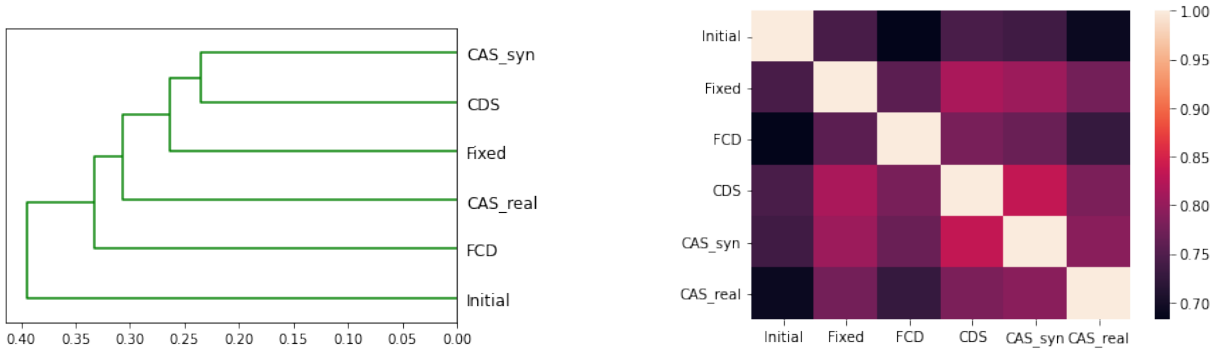

Figure 4.20: The Pearson correlation analysis of the minority class for all tabular datasets is represented using heatmap and dendrogram plots.

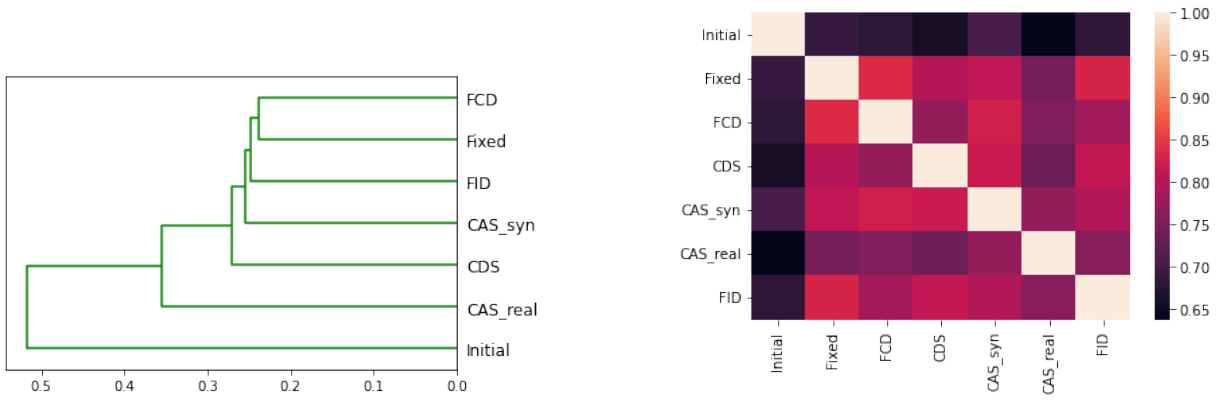

Figure 4.21: The Pearson correlation analysis of the minority class for all black-and-white imagery datasets (mnist-based, fmnist-based, and kmnist-based) is visualized through heatmap and dendrogram plots.

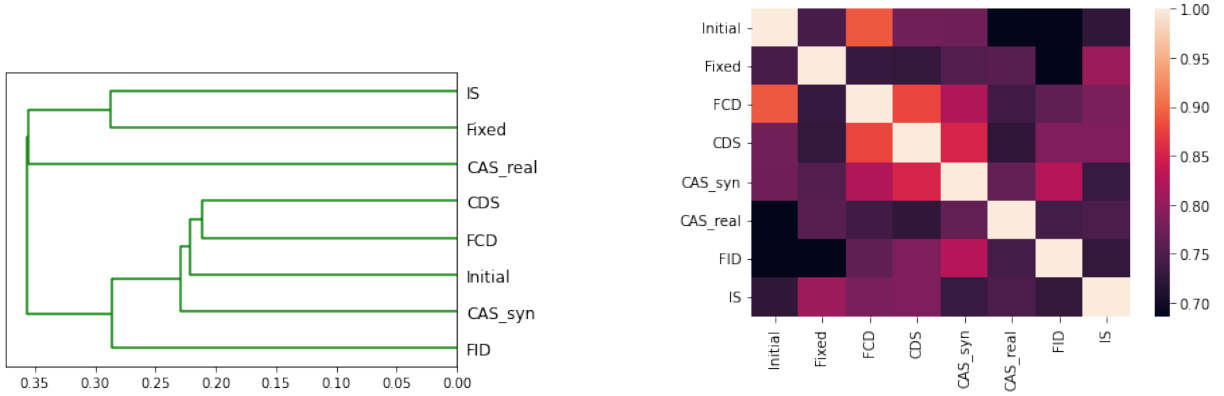

Figure 4.22: The Pearson correlation analysis of the minority class for color imagery datasets (cifar17-based) is represented using heatmap and dendrogram plots.

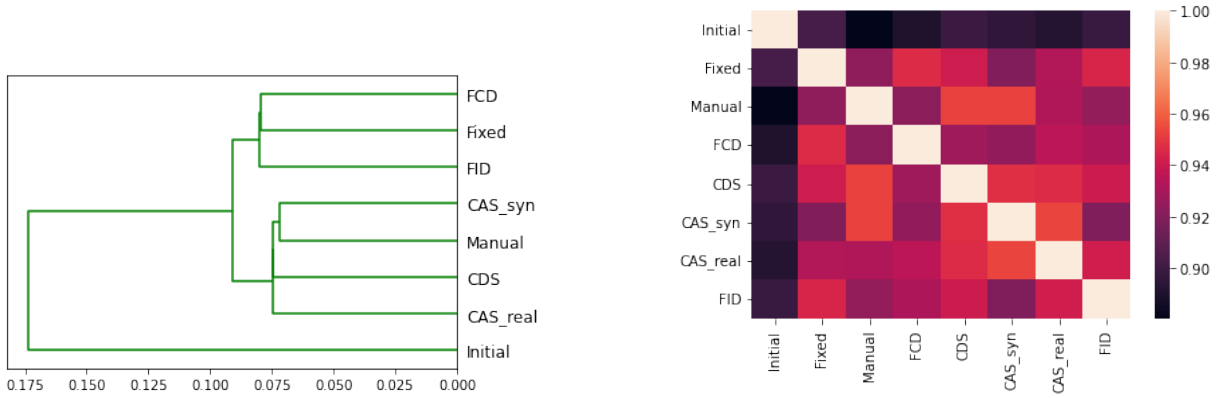

Figure 4.23: The Pearson correlation analysis of the minority class for fmnist-based datasets is visualized through heatmap and dendrogram plots.

4.6 Conclusions

In this chapter, we proposed a solution called AutoGAN that addresses the problem of determining when to stop training a GAN. AutoGAN is designed to automate the process and requires minimal human intervention. Through extensive experiments on various datasets, including both tabular and imagery data, we demonstrated that AutoGAN achieves comparable or better results than other methods. Additionally, AutoGAN reduces the number of training iterations required, achieving desirable outcomes more efficiently. Notably, when compared to manual human inspection, AutoGAN consistently stops training at earlier stages while maintaining high-quality generated data.

In terms of our findings on tabular datasets, it is noteworthy that the performance of the majority class remains largely unaffected, while the performance of the minority class improves or remains comparable. The performance outcomes of the AutoGAN algorithm are relatively comparable to those obtained by training with a fixed number of iterations, but AutoGAN algorithm achieves these results with significantly fewer training iterations.

Similar findings are obtained when analyzing the imagery data, even though a wider range of oracles and alternatives for stopping the GAN's training process were explored. In general, the oracles tested with AutoGAN yielded competitive results compared to training the GAN for a fixed number of iterations and also compared to the manual human inspection approach. These results are significant, indicating that human inspection, while feasible, is a time-consuming and subjective process that necessitates more training iterations to achieve the desired outcome.

In conclusion, our investigation of the Pearson correlation among the different methods reveals that the initial method exhibits the lowest correlation with the other methods overall. The correlation between the oracles and the other methods shows significant variability depending on the datasets examined. As a result, no definitive conclusion regarding the correlation of the other methods can be drawn universally.

4.7 Future Works

In terms of future research directions, our proposed framework paves the way for extending various GAN applications from the image domain to the tabular data domain, eliminating the need for supervision during GAN training. An intriguing avenue for future investigation could explore the transferability of image-to-image translation techniques to tabular-to-tabular translation or apply image de-noising methods to tabular data de-noising. AutoGAN can facilitate these novel tasks by enabling GAN training for tabular data without human intervention. Additionally, addressing issues such as mode collapse and vanishing gradients using AutoGAN presents another relevant research aspect. Our initial experiments reveal that GANs suffering from mode collapse exhibit significant score oscillations, while such behavior is absent when the mode collapse problem is absent. Furthermore, detecting overfitting in the GAN's generator could be another area of future exploration. We conjecture that specific oracle instances and/or modifications to the AutoGAN algorithm would be necessary to achieve this objective. Lastly, there is room to explore other methods of correlation measurement beyond the Pearson correlation, which only captures linear relationships. Exploring methods that can detect non-linear relationships would provide a deeper insight into the different methods tested.

Chapter 5

Enhancing Predictive Modeling: One Face to Rule Them All - Generalized Attacks Against Face Verification Systems

Deep neural network models have significantly advanced FV systems, leading to their use in applications like border control and smartphone unlocking. Despite their accuracy, these systems can be deceived by Adversarial Attacks, which subtly alter input images in ways typically imperceptible to humans. This chapter introduces a new attack called DodgePersonation Attack, offering a taxonomy of different attack types, and introducing a novel algorithm to implement the DodgePersonation Attack.

5.1 Introduction

FV is the process of determining if two face images belong to the same individual [80]. DeepFace, a Convolutional neural network model introduced in 2014 [132], achieved accuracy comparable to humans in FV using the Labeled Faces in the Wild (LFW) dataset [54]. Since then, neural network models have not only surpassed human performance in FV but have also advanced to the extent that these technologies are now employed in various public safety applications, such as border control procedures, as well as commercial applications like unlocking smartphones.

In ideal circumstances, these models exhibit impressive accuracy. However, their reliability is questionable when confronted with malicious intent. These models tend to produce incorrect predictions when confronted with imperceptible or seemingly natural adversarial input images [136]. These attacks can be categorized into two types: (1) Physical Attacks, where the model's input is manipulated (e.g., printing a photo or creating a 3D printed face to present to the FV system), and (2) Digital Attacks, where digital images are manipulated to deceive the FV system [136]. This chapter focuses on a specific kind of Digital Attack known as an Adversarial Attack, which alters a face image in a way that remains undetectable to the human eye, yet the FV system fails to identify the correct identity. These attacks are imperceptible to humans and can pave the way for the development of Physical Attacks, posing a significant risk.

Various strategies have been developed to trick FV systems. For example, Dodging Attack involves modifying a face image in a way that the FV system fails to recognize it as the original face, while still appearing indistinguishable from the original image to the human eye [16,65]. Another technique is called the Impersonation Attack, where the face image of individual A is manipulated to be identified as a different desired individual B, while still appearing as individual A to human observers [101]. A third attack scenario, known as the Master Faces or Master Face Attack, aims to generate images that can impersonate a wide range of identities [93,94,121]. In this study, we present a comprehensive analysis of these attacks, demonstrating that other types of attacks with significant risks for FV systems are also possible. Additionally, we introduce a novel attack framework that efficiently targets specific types of these attacks, surpassing existing methods in performance.

We present a generalized attack definition called the DodgePersonation Attack, which encompasses the Dodging, Impersonation, and Master Face Attacks, as well as introduces novel attack strategies. The DodgePersonation Attack is defined as follows: we select a set of human face images referred to as the $\mathcal{M}$atchSet, and another separate set called the $\mathcal{D}$odgeSet. Our primary goal is to generate one or multiple input face images that, as a collective, effectively impersonate all identities in the $\mathcal{M}$atchSet, while successfully evading recognition as any of the identities in the $\mathcal{D}$odgeSet.

Additionally, we propose a specific approach for executing the DodgePersonation Attack, which grants us control over the visual identity of the generated Attack Faces, a capability absent in previous research. Additionally, we strive to ensure that these input face images remain imperceptible to the human eye, appearing as a single image of the same person. Notably, our results for the Master Face Attack scenario significantly outperform previous work [121]. Moreover, we address new attack scenarios arising from the DodgePersonation Attack using the same framework.

Our study makes significant contributions in the following areas:

- A comprehensive understanding of attacks against FV systems is achieved by introducing the DodgePersonation Attack, a versatile attack that integrates various types of attacks.
- A taxonomy is proposed to categorize Adversarial Attacks against FV systems based on their specific targets.
- The novel algorithm, named "One Face to Rule Them All," is introduced to deploy the DodgePersonation Attack on FV systems. This algorithm achieves state-of-the-art results while allowing precise control over the identity of the generated Attack Faces.
- In a previous study, the Master Face Attack was addressed, achieving coverage of 43.82% of the dataset using a set of nine images, without imposing any limitations on the identity of the Attack Faces. In contrast, our method significantly enhances coverage to over 57% using the same number of images, while also meeting the additional requirement of utilizing a user-provided image for the attack.

In the remaining part of this chapter, the relevant literature in this field is examined in Section 5.2, followed by the introduction of the DodgePersonation Attack in Section 5.3, and the presentation of our new "One Face to Rule Them All" approach in Section 5.4. The experimental configurations are discussed in Section 5.5, while the outcomes are presented and analyzed in Section 5.6. Finally, the chapter is concluded in Section 5.8, and several future research directions are suggested in Section 5.9.

5.2 Related Work and Problem Motivation

5.2.1 Face Verification Systems

In general, automatic face recognition can be categorized into two primary groups: face identification, also known as closed-set face recognition, and FV. Face identification involves classifying faces into a predetermined set of identities, whereas FV determines whether two unseen identities, not encountered during the training phase, represent the same person [2, 26, 80]. As a result, FV is considered a zero-shot learning task, as the identities of individuals are unknown during the training phase [2]. Consequently, FV presents a higher

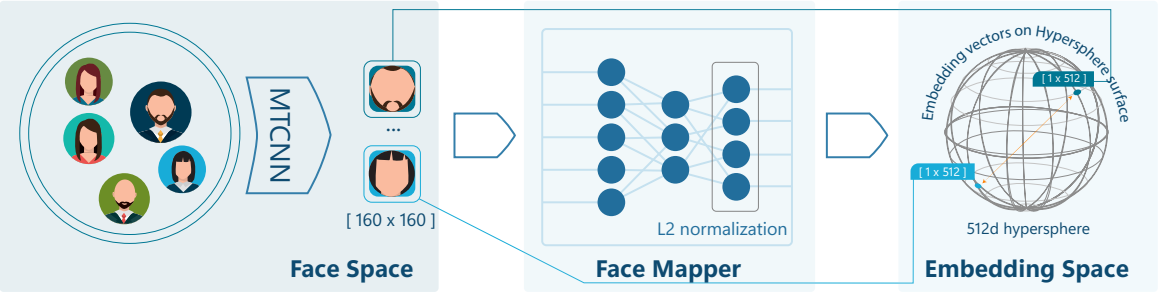

Figure 5.1: The FV process includes three steps: automatic face detection in images using MTCNN, mapping faces to embedding space with a neural network FM, and matching faces by comparing embeddings against a pre-defined threshold.

level of difficulty and complexity compared to face identification. This study focuses on attacking a FV system.

The current state-of-the-art solutions for FV, as illustrated in Figure 5.1, employ a three-step process. The initial step involves conducting face detection, such as employing the Multi-task Cascaded Convolutional Neural Networks (MTCNN) [150], which enables automatic detection of faces within a provided image. Secondly, a FM also known as a face descriptor is utilized to map a face from the image space to a corresponding embedding in a high-dimensional embedding space. The embedding represents the face's distinctive features and is typically L2-normalized. Neural network-based FMs are commonly employed in modern FV systems. Finally, in the face matching step, the distance between the embeddings of two faces is calculated, often using metrics like the Euclidean distance. If the distance falls below a predetermined threshold, the two faces are considered to belong to the same identity. Conversely, if the distance exceeds the threshold, the faces are regarded as belonging to different individuals or identities.

The choice of the loss function plays a crucial role in shaping the embedding space of a face recognition system. DeepFace [132], an early solution, utilized a vanilla softmax loss function. In addition to the aforementioned three-step process for face verification, DeepFace also incorporated a face frontalization step following face detection. This approach achieved an accuracy of 97.35% on the LFW dataset, comparable to the accuracy achieved by humans (97.53%). Another early method, DeepID2 [127], and its extension DeepID2+ [128] enhanced the performance by combining softmax loss with a distance metric. Their objective was to minimize the distance between pairs of the same identity and maximize the distance between pairs of different identities. These methods achieved accuracy rates of 99.15% and 99.47% respectively on the LFW dataset. A groundbreaking

advancement came with the introduction of FaceNet [116], which adopted a triplet loss function for direct learning of the embedding space. This approach further improved the performance, achieving an accuracy of 99.63% on the LFW dataset. Subsequently, several methods were proposed to enhance FaceNet. One such method [124] extended the triplet loss by allowing joint comparisons among multiple negative examples. Other approaches, such as SphereFace [75], CosFace [23, 139, 141], and ArcFace [22], leveraged a combination of softmax loss and marginal penalty loss to enhance the system's performance.

5.2.2 Attacks on Face Verification Systems

One of the primary challenges in FV is ensuring the robustness of the methods against various forms of attacks. These attacks encompass both physical and digital techniques. Examples of physical attacks involve modifying a face's appearance through the use of masks or makeup, whereas examples of digital attacks include manipulating images or videos [136]. Adversarial attacks, a subset of digital attacks, involve introducing small perturbations to an image with the intention of deceiving a model. In the context of face verification, adversarial attacks can be categorized as non-targeted or targeted attacks [136]. Non-targeted attacks, also known as Dodging Attacks, aim to cause the FV system to fail in recognizing the correct identity. Research in this field has focused on developing methods to generate such attacks, as exemplified by studies such as [27, 40, 118]. On the other hand, targeted attacks, also known as Impersonation Attacks, aim to make the FV system misidentify a specific individual. The research community has also worked on generating targeted attacks, including works such as [16, 27, 40, 118, 154]. Furthermore, a recent type of adversarial attack called the Master Face Attack has emerged. This attack aims to make the FV system accept the input as a match for all identities, effectively serving as a face master-key (also referred to as a Master Face). In [2], such images are created by minimizing the distance between the embedding of a perturbed image and a mini-batch of dataset embeddings. The authors take an arbitrary face image as input and craft the attacked image based on it. Additionally, three recent works have been published that generate Master Faces by exploring the latent space of a pre-trained GAN [93, 94, 121]. Unlike in [2], these works lack control over the identity of the synthesized Master Face, meaning the attack is not tailored to a specific chosen identity. Although classified as digital attacks, they do not fall under the category of adversarial attacks since they do not modify a given face image by adding perturbations. Instead, they generate a face that does not exist in reality with the assistance of GANs.

The existing literature on FV systems and attacks reveals gaps and unresolved issues. Previous research primarily concentrates on individual attack types, overlooking potential

connections between them. This study endeavors to tackle this by presenting a comprehensive perspective on various attacks, consolidating them into a unified problem. As a result of this new framework, previously unexplored attack scenarios have emerged. Furthermore, a novel and adaptable method is introduced, enabling the execution of diverse attack scenarios with success rates that outperform current state-of-the-art approaches.

5.2.3 Problem Motivation

As previously mentioned, the existing literature primarily concentrates on examining individual types of attacks in isolation, neglecting the potential inter-connectivity between them. This narrow focus has impeded the development of a comprehensive understanding of the broader landscape of attacks.

DodgePersonation Attack. To fill this void, we propose an attack formulation that combines different attack types, providing a unified perspective on attacking scenarios in FV. This formulation brings together a diverse range of attacks that were previously addressed individually, defining them under a single framework where each instance represents a unique manifestation of an attack. Our approach involves utilizing two distinct sets of face images, referred to as the $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet. The main objective is to deceive the FV system by presenting one or multiple inputs that are collectively recognized as matches for individuals in the $\mathcal{M}$atchSet, while simultaneously avoiding identification as any member of the $\mathcal{D}$odgeSet.

The concept of matching a set of identities is typically more intuitive compared to the need to evade another set of identities. We provide an illustration to demonstrate the relevance of the latter scenario based on a recent news case. Allegedly, a company used facial recognition technology to prevent a lawyer from entering their venues due to the lawyer's firm representing clients engaged in litigation against the company[1]. This incident exemplifies an attack scenario where our proposed system would effectively overcome such restrictive measures. In this particular situation, the lawyer facing denial of access could input their own face into our system and include it in the $\mathcal{D}$odgeSet. Consequently, the system would generate an output image resembling the attacker's face, yet it would remain unrecognized by the FV system. Furthermore, the targeted lawyer has the option to incorporate not only their own face but also the faces of their colleagues in the $\mathcal{D}$odgeSet. This approach serves the purpose of concealing their own identity and also prevents them from being identified as any other individuals who they suspect may also be denied access.

[1]Source: MSG probed over the use of facial recognition to eject lawyers from show venues

The concept of the **Source Face** is introduced, representing the face image provided by the user to initiate the attack. To execute the attack, multiple variations of the Source Face, referred to as **Attack Faces**, are generated with the goal of deceiving the FV system and satisfying the attack constraints. These modified images are intentionally crafted to be almost imperceptible to humans. The minimum number of Attack Faces required from a given Source Face depends on the specific attack scenario (discussed in Section 5.3.2) and the quality and performance of the employed FM. For example, in a Master Face Attack, a more advanced FM requires a larger number of Attack Faces to cover the majority of the provided identities. Through practical observations, it has been discovered that even with a relatively small number of Attack Faces, a significant majority of the identities can be encompassed. Further detailed explanations on this topic will be provided in subsequent sections.

5.2.4 Threat Model

5.2.4.1 The Goal of the Attacker

The primary aim of the attacker is to produce one or multiple face images that can be recognized as matching identities within the $\mathcal{M}$atchSet, all while avoiding detection as any individual from the $\mathcal{D}$odgeSet. In addition to this main objective, the attacker may have secondary goals, such as creating face images that are visually indistinguishable or resemble specific pre-selected images.

5.2.4.2 Capabilities and Limitations of the Attacker

In this investigation, a white-box attack [136] scenario is examined, granting the attacker unrestricted access to the target model. Nevertheless, the attacker's capabilities are constrained by the need to generate one or more face images (Attack Faces) that can effectively bypass the face detection step in the FV process. Therefore, the attacker must create images that fulfill their objective while maintaining the appearance of human faces. Additionally, there may be limitations on the attacker's ability to craft the face images to closely resemble a specific user-input identity, known as the Source Face.

5.2.4.3 Possible Attack Scenarios

Different attack scenarios arise based on the composition of the $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet populations. For example, consider a scenario where a criminal, acting as an attacker,

aims to fraudulently apply for a license online by impersonating someone else and avoiding recognition. In this situation, the objective of the attacker is to produce one or multiple images that, as perceived by the FV system, resemble the face of the victim while being perceived as the face of the attacker by humans. This scenario can be represented by including the victim's face image in the $\mathcal{M}$atchSet and the attacker's face image in the $\mathcal{D}$odgeSet, where the attacker's face image is used as the Source Face. Further scenarios are elaborated on in Section 5.3.2.

5.2.4.4 Targeted Attack Model

The FV system we target for our attack is FaceNet, which is a neural network-based system.

5.3 The *DodgePersonation Attack*

In this section, a taxonomy for FV system attack scenarios is presented. The terms and notations are defined, followed by the introduction of the mathematical definitions for impersonation and dodging of face images. Two sets of face images, namely $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet, are introduced, which play a crucial role in the DodgePersonation Attack. These sets form the foundation for problem definition and enable the categorization of different attacks within the proposed taxonomy.

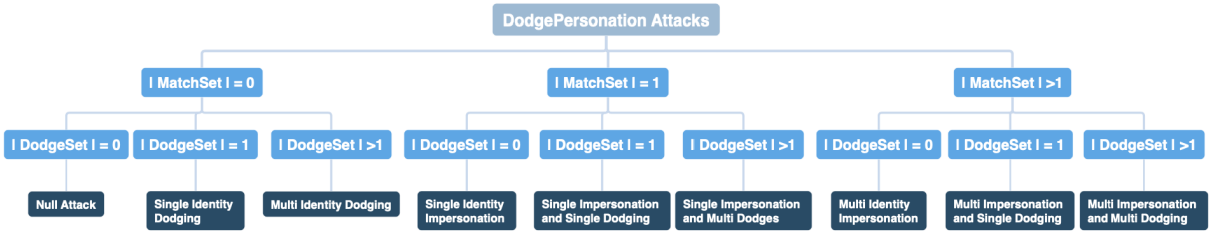

Figure 5.2: The proposed DodgePersonation Attack Taxonomy is introduced to categorize various attack scenarios targeting FV systems.

5.3.1 DodgePersonation Attack Definition and Taxonomy

Assume that $\mathcal{F}$ represents the collection of all face images, and $\mathcal{I}$ represents the set of all identities. Let us define a function $Ident() : \mathcal{F} \rightarrow \mathcal{I}$ that associates a face image with its

corresponding identity. For example, if we have three face images i_1, i_2, i_3 that belong to the same identity $I_1 \in \mathcal{I}$, we can observe that $Ident(i_1) = Ident(i_2) = Ident(i_3) = I_1$.

Let us consider a face image represented by three squared matrices, denoted as $M = m \times m$, where each matrix represents a color channel. We have a given FV system that can be divided into two components: (i) a FM function, denoted as $FM()$; and (ii) a distance function, denoted as $Dist()$. The purpose of the FM function, $FM() : M^3 \rightarrow \mathbb{R}^p$, is to project a face image into an embedding space. When a single image x (or a set of images S) is mapped from the face image space to the embedding space, it is represented as $\bar{x}$ (or $\bar{S}$) to distinguish it from the original representation. In the embedding space, all objects are denoted with a bar. The distance function, $Dist() : \mathbb{R}^p \rightarrow \mathbb{R}$, is used to calculate the distance between two points in the embedding space, serving as a hard measure of similarity. To determine whether the identities of two face images match or mismatch in the embedding space, a threshold value $th \in \mathbb{R}$ is employed. As mentioned in Section 5.2, the FM function, $FM()$, utilizes a neural network architecture to extract features from the images.

Definition 5.1 (Impersonation and Dodging)**.** *Let $A, B \in \mathcal{F}$. Face Image A is considered to **impersonate** Face Image B when:*

$$Dist(FM(A), FM(B)) \leq th \tag{5.1}$$

*If the condition is not satisfied, we describe Face Image A as **dodging** Face Image B.*

Definition 5.2 ($\mathcal{M}$atchSet and $\mathcal{D}$odgeSet Face Image Sets)**.** *Consider $\mathcal{M}atchSet = \{m_1, m_2, \cdots, m_{k-1}, m_k\} \subset \mathcal{F}$ as a set of images, and $\mathcal{D}odgeSet = \{d_1, d_2, \cdots, d_{l-1}, d_l\} \subset \mathcal{F}$ as another set of images. Throughout the remainder of the chapter, we will assume that sets of face images in a given problem adhere to certain conditions. Specifically, these sets must satisfy the following criteria:*

- $k \geq 0$
- $l \geq 0$
- $\mathcal{M}atchSet \cap \mathcal{D}odgeSet = \emptyset$

With the establishment of $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet, we are now able to define the DodgePersonation Attack and classify the various attacks within a taxonomy.

Definition 5.3 (DodgePersonation Attack)**.** *Let us consider two sets, $\mathcal{M}atchSet$ and $\mathcal{D}odgeSet$. The DodgePersonation Attack can be defined as a multi-objective optimization problem, aiming to generate a collection of images called "Attack Faces", denoted as $\mathbf{X} = x_1, x_2, \ldots, x_n$. These images need to meet the requirement of being recognized as faces by the MTCNN face detector, allowing them to proceed successfully to the subsequent steps of face verification. For each $x_i \in X$, we define M_{x_i} and D_{x_i} as follows:*

$$\begin{cases} M_{x_i} := \{m \in \mathcal{M}atchSet : Dist(FM(x_i), FM(m)) \leq th\} \\ D_{x_i} := \{d \in \mathcal{D}odgeSet : Dist(FM(x_i), FM(d)) \leq th\} \end{cases} \tag{5.2}$$

Subsequently, our multi-objective optimization problem can be formulated as follows:

$$\begin{cases} \max|\bigcup_{\forall x_i \in X} M_{x_i}| \\ \min|\bigcup_{\forall x_i \in X} D_{x_i}| \end{cases} \tag{5.3}$$

5.3.2 DodgePersonation Attack taxonomy

Based on the size of $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet, the DodgePersonation Attack can be categorized into multiple types of attacks. These attack types are presented in our proposed taxonomy, illustrated in Figure 5.2. The first layer of the taxonomy focuses on the number of identities to be impersonated, corresponding to different sizes of $\mathcal{M}$atchSet. At this level, we consider three scenarios: (1) no impersonation, (2) impersonating a single identity, and (3) impersonating multiple identities. Moving to the second layer, we consider the number of identities to be dodged from, which is related to the size of $\mathcal{D}$odgeSet. This layer consists of three branches: (1) no identity to dodge from, (2) dodging a single identity, and (3) dodging multiple identities. By combining the desired number of identities to impersonate and dodge, a total of nine possible scenarios emerge from this taxonomy.

Scenarios in which the $\mathcal{M}$atchSet size is zero and $\mathcal{D}$odgeSet size is one are commonly known in the research community as None-Targeted Attack, Face De-Identification, or Dodging Attack. On the other hand, scenarios where the $\mathcal{M}$atchSet size is one and $\mathcal{D}$odgeSet size is zero are referred to as Targeted Attack, as mentioned in previous literature [136]. When the $\mathcal{M}$atchSet size is large, indicating the presence of numerous identities, and the $\mathcal{D}$odgeSet size is zero, it is denoted as Master Faces or Master Face Attack [94]. Additionally, our proposed taxonomy incorporates novel scenarios that have not yet been established in the literature and therefore lack specific names.

By examining FV systems from the standpoint of an attacker, we can investigate the circumstances in which these systems can be tricked. The goal of the attacker is to generate a collection of images that visually resemble them (with the Source Face being an image of the attacker) but are recognized by the FV system as belonging to a different individual or individuals. Here, we provide a compilation of practical situations that align with particular instances outlined in our taxonomy, as illustrated in Figure 5.2.

Null Attack. The first scenario, referred to as the Null Attack, arises when there is no objective to impersonate or dodge any specific identity. In this straightforward case, any valid input serves as a solution.

Single Identity Dodging. In one attack scenario, an individual seeks to protect their identity from being recognized by an online FV system employed on social media. Their objective is to evade the system's facial identity verification mechanism by generating an image that does not resemble their own face. This scenario can be achieved by forming an empty $\mathcal{M}$atchSet and a $\mathcal{D}$odgeSet containing the attacker's own face image. In the research community, this scenario is commonly referred to as the None-Targeted Attack, Face De-Identification, or Dodging Attack [136].

Multi Identity Dodging. In this scenario, the attacker, who is a wanted criminal, has the objective of concealing their identity and ensuring that the modified image does not bear any resemblance to other potentially sensitive identities, such as other criminals. To reduce the risk of being identified as either themselves or another criminal, the attacker needs to generate an image that differs from multiple identities, including their own. This can be accomplished by creating an empty $\mathcal{M}$atchSet and a $\mathcal{D}$odgeSet comprising the images associated with those identities.

Single Identity Impersonation. In an alternate attack scenario, an individual may aim to gain unauthorized access to someone else's smartphone by assuming the identity of the victim [2]. To accomplish this, the attacker must generate a collection of images that closely resemble the victim's face. This can be expressed as a $\mathcal{M}$atchSet consisting of a single member, which is the face image of the victim, accompanied by an empty $\mathcal{D}$odgeSet. The research community refers to this scenario as a Targeted Attack [136].

Multi Identity Impersonation. This scenario may occur when an attacker intends to deceive an online portal's FV system in order to gain unauthorized access. In this situation, the attacker lacks complete information about the authorized employees, including which individuals possess access privileges and which do not. To maximize their chances of

[2]For this particular scenario, we assume direct access to the smartphone's FV system. In future research, exploring Physical Attacks would represent a logical progression, as they present a more realistic setting in this context.

success, the attacker must impersonate all potential individuals who might have proper access. This can be achieved by creating a $\mathcal{M}$atchSet consisting of multiple members, which correspond to the face images of the employees. Similarly, in another scenario, the attacker may seek to gain unauthorized access to any random smartphone by tricking its FV system. To accomplish this, the attacker must generate a collection of images that resemble a large number of identities. In this case, the scenario can be formulated by establishing a $\mathcal{M}$atchSet that contains face images representing a significant number of identities, while leaving the $\mathcal{D}$odgeSet empty. This scenario represents an extreme case of the previous scenario and is commonly referred to as a Master Face Attack [94].

Single Impersonation and Single Dodging. Another scenario emerges when there is a simultaneous requirement for both **Single Identity Dodging** and **Single Identity Impersonation**. To illustrate this scenario, let us consider a situation where a wanted criminal, acting as an attacker, aims to exploit an online system equipped with an FV system by using their own image to gain unauthorized entry. In this case, the attacker intends to access the system by impersonating an authorized individual while ensuring their own identity remains unrecognized. To achieve this, the attacker must generate a set of images that match the face of the victim but do not resemble their own face. This scenario can be represented by a $\mathcal{M}$atchSet containing the face image of the victim as the sole member, and an $\mathcal{D}$odgeSet consisting of the attacker's own face image. Similarly, in another scenario known as **Single Impersonation and Multi Dodging**, the attacker aims to conceal their identity and prevent the modified image from resembling any other critical identities. Moreover, in situations where the attacker has limited information about the individuals they wish to impersonate, including uncertainty about who has access privileges, and also needs to avoid being recognized as themselves, the scenario of **Multi-Impersonation and Single Dodging** arises.

Multi Impersonation and Multi Dodging. Lastly, consider a scenario where an organization grants access to specific individuals but alerts the police if certain wanted individuals are identified. In this case, the attacker's objective is to impersonate the authorized individuals and prevent their own recognition as one of the wanted individuals. To maximize their chances of gaining access while minimizing the risk of being apprehended, the attacker can employ a $\mathcal{M}$atchSet comprising face images of multiple authorized individuals who potentially have access privileges. Simultaneously, they can use a $\mathcal{D}$odgeSet consisting of face images of the wanted individuals. By adopting this approach, the attacker can decrease the likelihood of being identified as a wanted individual while increasing their likelihood of successfully entering the premises.

5.4 One Face to Rule Them All Method

This section introduces the **One Face to Rule Them All Algorithm**, which is utilized for conducting DodgePersonation Attack. Our algorithm comprises two main phases: Phase 1 - the embedding space search, and Phase 2 - the Attack Face generation. In Phase 1, we employ the FM function to map the $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet to the embedding space $\mathcal{E}$. To search for a point in the embedding space that satisfies the attack's constraints, we utilize a GA with a specialized fitness function. During Phase 2, we modify the user's input face image (referred to as the Source Face) to ensure that when it undergoes mapping to the embedding space using FM, it is positioned in close proximity to the point discovered in Phase 1. This ensures that the manipulated user input image adheres to the constraints set by the $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet.

It is important to note that our algorithm imposes an additional constraint on the DodgePersonation Attack outlined in Definition 5.3. We require that the set of Attack Faces generated by our algorithm, i.e., the manipulated images, bear a resemblance to a single predetermined identity (the Source Face) as perceived by humans. This means that while the Algorithm allows for the use of different Source Faces, we specifically focus on a special case where all Attack Faces are crafted from a single user-input Source Face. We deliberately limit ourselves to a Single Source Face for all Attack Faces to showcase the algorithm's capability in producing face images that are nearly indistinguishable from the Source Face to the human eye. Consequently, this constraint adds complexity to the attack generation process compared to scenarios without this restriction.

5.4.1 Phase 1 - Embedding Space Search

The objective of Phase 1 is to identify one or more points in the embedding space that correspond to the desired Attack Face(s) we aim to find. To accomplish this, we employ the FV system to map the $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet from the face space to the embedding space. We utilize a bar notation to represent the elements in the embedding space, resulting in $\overline{\mathcal{M}\text{atchSet}}$ and $\overline{\mathcal{D}\text{odgeSet}}$. To determine the desired points, we propose two key steps: (i) constructing clusters based on $\overline{\mathcal{M}\text{atchSet}}$; and (ii) conducting a search in the embedding space around each cluster using a GA combined with a specially designed fitness function. Figure 5.3 provides an overview of Phase 1, illustrating these steps.

From Face Space to Embedding Space. The face images in $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet undergo processing by the MTCNN algorithm and are pre-processed for the FM function. Finally, they are fed into the FM function, as illustrated in Figure 5.1.

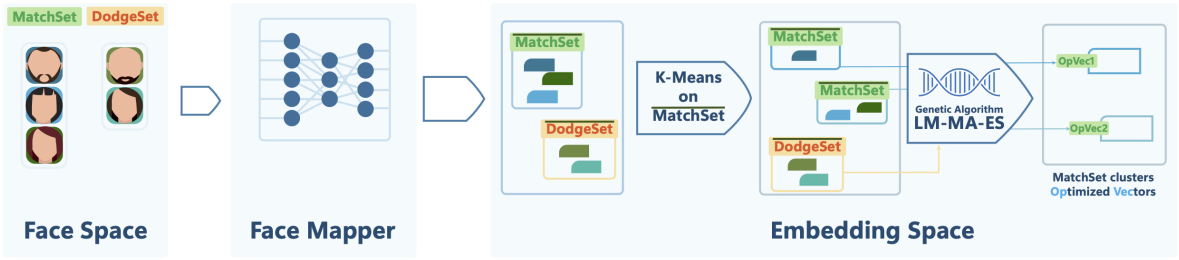

Figure 5.3: Overview of One Face to Rule Them All - Phase 1: Embedding Space Search. The $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet are mapped from the face space into the embedding space. Then, C clusters are built using $\overline{\mathcal{M}\text{atchSet}}$. For each cluster, one point is sought using our GA such that it is close to the cluster members and distant from the $\overline{\mathcal{D}\text{odgeSet}}$ members.

Cluster Generation. The ultimate goal is to identify a set of points, $\overline{x_i}$, within the embedding space $\mathcal{E}$ that satisfy two conditions: (i) collectively being close to all elements of $\overline{\mathcal{M}\text{atchSet}}$; and (ii) $\overline{x_i}$ being distant from all elements of $\overline{\mathcal{D}\text{odgeSet}}$. However, a single point may not fulfill these requirements since the elements in $\overline{\mathcal{M}\text{atchSet}}$ could be distributed across the embedding space. Hence, we generate C clusters of the $\overline{\mathcal{M}\text{atchSet}}$ cases ($\overline{\mathcal{M}\text{atchSet}_1}, \cdots, \overline{\mathcal{M}\text{atchSet}_C}$) by utilizing the K-Means algorithm [76]. The number of clusters serves as a hyper-parameter for the system.

We aim to address the multi-objective optimization problem for each cluster $\overline{\mathcal{M}\text{atchSet}_i}$, which can be formulated as follows:

$$\begin{cases} max|\{\overline{m} \in \overline{\mathcal{M}\text{atchSet}_i} : Dist(\overline{x}_i, \overline{m}) \leq th\}| & (5.4) \\ min|\{\overline{d} \in \overline{\mathcal{D}\text{odgeSet}} : Dist(\overline{x}_i, \overline{d}) \leq th\}| & (5.5) \end{cases}$$

We utilize the LM-MA-ES GA, as introduced in [78], to discover a suitable point within the high-dimensional embedding space that satisfies our criteria. This particular GA was chosen due to its effectiveness in exploring spaces with a high number of dimensions, as demonstrated in the study by [121]. To evaluate the effectiveness of a point in meeting our requirements, we minimize the DodgePersonation Fitness function (defined in Definition 5.5). This fitness function comprises two components, positive and negative, which correspond to the objectives stated in Equations 5.4 and 5.5. By employing the $DPloss$ (defined in Definition 5.4) on either the $\overline{\mathcal{M}\text{atchSet}_i}$ or $\overline{\mathcal{D}\text{odgeSet}}$, we derive the weighted sum of these two components within the DodgePersonation Fitness function.

The $DPloss$ quantifies the dissimilarity between a target case $\overline{a}$ and a given set $\overline{S}$. It

Algorithm 5.1 One Face to Rule Them All - Phase 1: Embedding space Search

Input: $\overline{\mathcal{M}\text{atchSet}}$: set of all face embeddings to impersonate;
$\overline{\mathcal{D}\text{odgeSet}}$: set of all face embeddings to dodge;
α, β: weights for positive and negative $DPloss$, respectively;
γ: weight for the DodgePersonation Fitness function
$th1$, $th2$: DodgePersonation Fitness thresholds
gen: number of GA generations
C: number of clusters
Output: $best\_emb$: best C points found in the embedding space.

$best\_emb \leftarrow []$
$Clusters \leftarrow K\text{-}Means(\ C,\ \overline{\mathcal{M}\text{atchSet}})$
for *cluster in Clusters* **do**
  $centroid_{norm} \leftarrow$ cluster centroid normalized to unit vector
  $pp \leftarrow$ initialize first population with $centroid_{norm}$
  for *generation in gen* **do**
    $pp' \leftarrow$ generate new population based on pp
    $pp'_{norm} \leftarrow$ normalize to unit vector elements of pp'
    $loss \leftarrow []$
    for pp'_i *in* pp'_{norm} **do**
      $loss \leftarrow loss \cup fitness(pp'_i, cluster, \overline{\mathcal{D}\text{odgeSet}}, th1, th2, \alpha, \beta, \gamma)$
    end
    $pp \leftarrow$ top $|pp'|$ with lowest $loss$ from pp'_{norm}
  end
  $best\_emb \leftarrow best\_emb \cup$ embedding with lowest loss in pp
end
return $best\_emb$

considers two aspects: the count of cases in $\overline{S}$ that are farther away from $\overline{a}$ than a specified threshold, and the sum of the distances between $\overline{a}$ and all the cases in $\overline{S}$.

Definition 5.4 (DP Loss)**.** *loss is a normalized linear combination of* $DodgeCount(\overline{a}, \overline{S}, th)$ *and* $DistSum(\overline{a}, \overline{S})$*:*

$$DPloss(\overline{a}, \overline{S}, th, m) = \frac{m \cdot \sum_{\overline{s} \in \overline{S}} \mathbb{1}_{\{Dist(\overline{a},\overline{s}) > th\}}(\overline{a}) + (1-m) \cdot \sum_{\overline{s} \in \overline{S}} Dist(\overline{a}, \overline{s})}{|\overline{S}|}$$

where the weight factor m lies within the range of [0,1], and the indicator function $\mathbb{1}_{Dist(\overline{a},\overline{s})>th}(\overline{a})$ *evaluates to 1 if the condition* $Dist(\overline{a}, \overline{s}) > th$ *is true, and 0 otherwise.*

To eliminate the influence of the size of $\overline{S}$ on the outcome, we normalize the result. In our preliminary experiments, we found that relying solely on the indicator function component was inadequate for the GA to effectively choose appropriate individuals for the next generation. To address this issue, we resolved it by incorporating the distances between the searched point and the members of $\overline{S}$.

The fitness function used in our GA, called the DodgePersonation Fitness Function, is determined by the weighted sum of the positive $DPloss$ applied to each cluster $\overline{\mathcal{M}\text{atchSet}_i}$ and the negative $DPloss$ (multiplied by -1) applied to the $\overline{\mathcal{D}\text{odgeSet}}$. This definition can be found in Definition 5.5.

Definition 5.5 (DodgePersonation Fitness Function)**.** *Considering a point* $\overline{x}$ *in the embedding space* $\mathcal{E}$*, we examine a cluster* $\overline{\mathcal{M}atchSet_i}$ *from* $\overline{\mathcal{M}atchSet}$ *and the* $\overline{\mathcal{D}odgeSet}$*. Decision thresholds* $th1$ *and* $th2$ *are established for* $\overline{\mathcal{M}atchSet_i}$ *and* $\overline{\mathcal{D}odgeSet}$ *respectively. We introduce weights* α *and* β *for the DPloss components, along with a weight parameter* γ *for the fitness function. The DodgePersonation Fitness function is defined as follows:*

$$\begin{aligned} fitness(\overline{x}, \overline{\mathcal{M}atchSet_i}, \overline{\mathcal{D}odgeSet}, th1, th2, \alpha, \beta, \gamma) = \\ \gamma * DPloss(\overline{x}, \overline{\mathcal{M}atchSet_i}, th1, \alpha) + \\ (1-\gamma) * (-DPloss(\overline{x}, \overline{\mathcal{D}odgeSet}, th2, \beta)) \end{aligned}$$

It is important to note that our DodgePersonation Fitness Function incorporates distinct thresholds for the positive and negative $DPloss$, enabling independent adjustments. This feature enhances the flexibility of the method and leads to more favorable outcomes.

The pseudo-code for Phase 1 of the One Face to Rule Them All algorithm, which involves searching for a point or set of points in the embedding space to carry out the DodgePersonation Attack, is presented in Algorithm 5.1.

5.4.2 Phase 2 - Attack Face Generation

After obtaining a set of points ($\overline{x_i}$) in the embedding space, the next step is to generate the corresponding Attack Face images. We begin with a Source Face x and aim to alter the image in a manner that, when processed through the FM, it is mapped closely to the Phase 1 point ($\overline{x_i}$). In other words, our objective is to ensure that the output of $FM(x_{modified})$ is as close as possible to $\overline{x_i}$.

A method was developed for the FV task that utilizes a pre-selected Source Face image x and minimizes the extent of modifications while ensuring alignment with the previously obtained point $\overline{x_i}$. The Attack Face Generation method operates by computing the derivatives of the FM function with respect to the input image x and modifying the image to reduce the loss value, represented by $Dist(FM(x), FM(\overline{x}_i))$. To the best of our knowledge, this mapping technique, which grants complete control over the initial identity in the face image, has not been previously applied in the domain of FV, making it a novel approach in this field.

Algorithm 5.2 One Face to Rule Them All Algorithm - Phase 2: Attack Face Generation.

Input: x: Source Face;
$\overline{y}$: point in the embedding space returned in Phase 1;
ϵ: maximum change allowed for each pixel of the normalized image;
$iterations$: number of iterations;
Output: x_{adv}: Attack Face
while $x \neq MTCNN(x)$ **do**
  $x \leftarrow MTCNN(x)$
end
$x_{adv} \leftarrow x$
for *iter in iterations* **do**
  $\overline{x}_{adv} \leftarrow FM(x_{adv})$
  $loss \leftarrow Dist(\overline{x}_{adv}, \overline{y})$
  $gradients \leftarrow \frac{\partial loss}{\partial x_{adv}}$
  $x_{adv} \leftarrow Adam(x_{adv}, gradients)$
  $x_{adv} \leftarrow clip(x_{adv}, [x_{adv} - \epsilon, x_{adv} + \epsilon])$
end
return x_{adv}

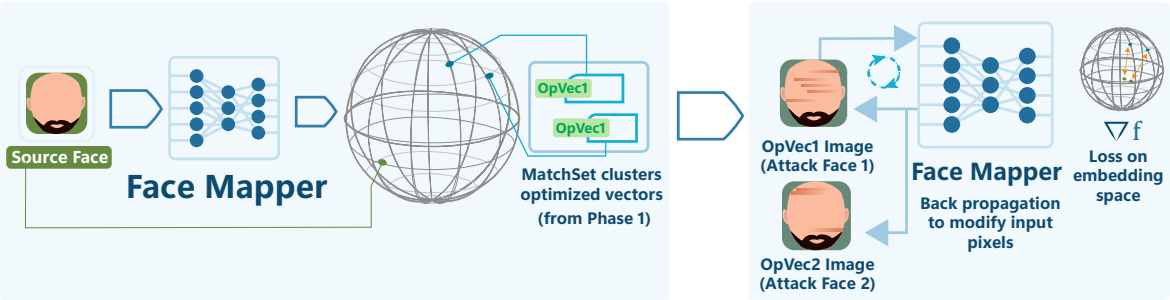

Figure 5.4: In Phase 2 of One Face to Rule Them All - Attack Face Generation, the Source Face image is transformed into the embedding space and adjusted to be close to a point identified in Phase 1. This approach allows us to launch attacks on various face images while maintaining control over the extent of modifications made.

The pseudo-code for the Attack Face Generation process is presented in Algorithm 5.2. Initially, the face is extracted from the Source Face x using the MTCNN face detector. This involves feeding x to MTCNN and resizing it to match the FM's input shape. The cropping continues until MTCNN no longer trims the input, ensuring the input and output dimensions are identical. Next, the cropped version of x undergoes a specified number of iterations for embedding the attack into the image.

Within each iteration, the algorithm computes the distance between the image mapped into the embedding space $FM(x)$ and the target point $\overline{x}$ in the embedding space. Additionally, the derivative of FM with respect to the image is calculated. While the Adam optimizer is commonly used to update neural network weights for loss minimization, we have repurposed it in an innovative manner to apply the obtained gradients to the image. We also introduce a constraint to control the magnitude of change, limiting the difference (ϵ) between the initial and modified images. This approach produces a final version of x that retains a high visual resemblance to the original cropped Source Face image. Figure 5.4 provides an overview of the One Face to Rule Them All Phase 2 process.

5.5 Experimental Setup

In this section, we assess the effectiveness of our approach in executing DodgePersonation Attacks through the utilization of the One Face to Rule Them All Algorithm. We start by presenting the dataset employed in our experiments and subsequently provide implemen-

tation specifics pertaining to our experimental setup.

5.5.1 Dataset

The experiments were conducted on the LFW dataset, which is a widely used dataset for FV tasks. This dataset consists of 13,233 images of 5,749 individuals. The funneled version of the dataset was used. The FV decision threshold for a false acceptance rate of 0.001 was obtained by applying Euclidean distance on the training set provided with LFW, which contains 1,100 matching and 1,100 mismatching pairs. Unless specified otherwise, a random subset of 5,749 images was used in the remaining experiments, where each individual in the dataset is represented by a single image. The dataset, along with the corresponding code, can be accessed from the GitHub repository.

5.5.2 Evaluation Metrics

To assess the effectiveness of our approach, we determine its performance by measuring the coverage score on a specific collection of images, denoted as set S, while considering a set of Attack Faces X. The coverage score is computed as follows: we count the number of face images in S that have a match with at least one image in the set of Attack Faces X, using a distance metric $Dist$ and a threshold value th. This count is then divided by the total number of images in S, and multiplied by 100 to express the result as a percentage. The calculation of the coverage is described more formally in Definition 5.6.

$$coverage = \frac{|\{s \in S : \exists x \in X,\ Dist(FM(x), FM(s)) < th\}|}{|S|} \tag{5.6}$$

The coverage score indicates the proportion of face images in S that are successfully matched by at least one image in the Attack Faces set X. We calculate the coverage for both the $\mathcal{M}$atchSet and the $\mathcal{D}$odgeSet, with the objective of maximizing it for the $\mathcal{M}$atchSet and minimizing it for the $\mathcal{D}$odgeSet.

5.5.3 Implementation Details

Based on the specified sizes of $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet, we selected two distinct sets of identities from the LFW dataset in a random manner.

We selected one image per identity. Alternatively, if we were to use multiple images for each identity and assume that FM effectively groups them in a nearby region of the embedding space, we would introduce additional closely located points to the Phase 1 Embedding Space Search. This simplifies the task for the search algorithm in finding a suitable point in the embedding space. Consequently, with this additional information, the algorithm is capable of generating more precise results. In contrast, when we choose to use only one image per identity, we are selecting a potentially more challenging scenario.

We utilized the default configurations of the MTCNN algorithm and adjusted the image size to match the input shape of the FM. This resizing was accomplished using the PIL library with the BOX re-sampling algorithm. The input shape was set to 160x160 pixels, and the input images were normalized from a range of 0 to 255 to a range of -1 to +1, as required by FaceNet [116], the specific FM employed in our approach. FaceNet is trained on sets of three faces (known as triplets) from the CASIA-WebFace dataset [147] using a triplet loss function. By leveraging FaceNet, we can map images from the original image space onto a compact Euclidean space. In this embedding space, each point is represented by a fixed-length vector of 512 dimensions.

During Phase 2, we employed the Euclidean distance as the loss function for mapping the output. To determine the appropriate number of training iterations for this phase, we conducted a series of initial experiments and determined that 1,000 iterations yielded satisfactory results.

5.6 Results and Discussion

In this section, we showcase the outcomes of our experiments aimed at evaluating the performance of our proposed One Face to Rule Them All Algorithm. These experiments were conducted across various scenarios as outlined in our taxonomy. Additionally, we provide two ablation studies that focus on examining the impact of specific parameters in our DodgePersonation Fitness function.

5.6.1 Results for $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet of varying size

In this set of experiments, the focus was on 10 clusters within the $\mathcal{M}$atchSet. Our solution was evaluated using different sizes for the $\mathcal{M}$atchSet, specifically 10, 100, 500, 1000, and 2500. Similarly, the $\mathcal{D}$odgeSet was examined across sizes of 0, 1, 2, 3, 10, 100, 1000, and 2500.

To create the $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet, elements were randomly selected from the LFW dataset. These sets were subsequently mapped into the embedding space, resulting in the formation of $\overline{\mathcal{M}\text{atchSet}}$ and $\overline{\mathcal{D}\text{odgeSet}}$. The One Face to Rule Them All Algorithm was then applied to identify a single point (Attack Face) within each cluster that adheres to the defined problem constraints.

We assessed the coverage of the generated points in the embedding space during Phase 1, specifically in $\overline{\mathcal{M}\text{atchSet}}$ and $\overline{\mathcal{D}\text{odgeSet}}$, as well as the coverage of the corresponding generated Attack Faces during Phase 2, in $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet. This evaluation process was repeated five times, and the average results are depicted in Figure 5.5. The coverage outcomes for Phase 1 and Phase 2 are presented in blue and orange, respectively.

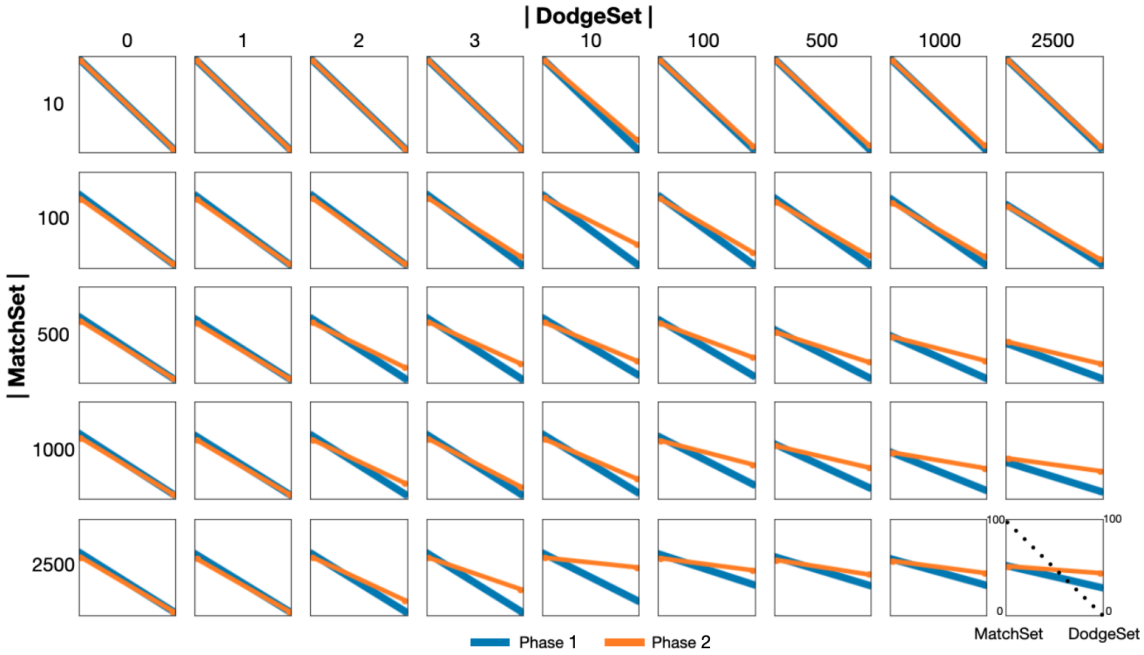

Figure 5.5: Coverage outcomes for varying $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet sizes are shown, with each plot depicting $\mathcal{M}$atchSet coverage on the left and $\mathcal{D}$odgeSet coverage on the right. Lines connecting both sets' results are color-coded according to the computation phase.

Each subplot within the figure represents a particular scenario, which corresponds to a combination of specific sizes for $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet. For instance, the bottom right subplot illustrates the coverage results for a $\mathcal{M}$atchSet with a size of 2,500 and a $\mathcal{D}$odgeSet with a size of 2,500. In each subplot, the left side represents the coverage on $\overline{\mathcal{M}\text{atchSet}}$

(Phase 1 in blue) and $\mathcal{M}$atchSet (Phase 2 in orange), while the right side represents the coverage on $\overline{\mathcal{D}\text{odgeSet}}$ (Phase 1 in blue) and $\mathcal{D}$odgeSet (Phase 2 in orange).

The optimal outcome for both Phase 1 and Phase 2 in each subplot is characterized by a coverage of 100% on the $\mathcal{M}$atchSet (left side) and a coverage of 0% on the $\mathcal{D}$odgeSet (right side). This ideal result is represented by a diagonal line extending from the top left corner to the bottom right corner of the subplot (shown as a dashed line in the bottom right subplot).

To illustrate, let us consider the subplot in which the $\mathcal{M}$atchSet has a size of 100 and the $\mathcal{D}$odgeSet has a size of 2,500. In Phase 1, we observe a coverage of 66.33% on $\overline{\mathcal{M}\text{atchSet}}$, indicating that a significant portion of the points in this set is covered. Conversely, only 0.13% of the $\overline{\mathcal{D}\text{odgeSet}}$ is covered during this phase.

Moving on to Phase 2, we find that the coverage on the $\mathcal{M}$atchSet is 65.33%, which is slightly lower compared to Phase 1. However, the coverage on the $\mathcal{D}$odgeSet increases to 5.16%. Notably, the lines representing the results of Phase 1 and Phase 2 are nearly overlapping, suggesting that the outcomes are quite similar. This demonstrates the effectiveness of Phase 2 of the One Face to Rule Them All algorithm, as it successfully generates Attack Faces of the selected identity that closely align with the embedding space point discovered during Phase 1.

In general, the experiments conducted with a smaller number of faces to match and dodge (subplots located close to the top left) yield results that are close to optimal. As the number of faces to match and dodge increases, we observe a decrease in the coverage of the $\mathcal{M}$atchSet and an increase in the coverage of the $\mathcal{D}$odgeSet. The scenario involving 2,500 faces to match and 2,500 faces to dodge (bottom right subplot) represents the most challenging and least successful scenario. However, even in this demanding situation, a coverage of approximately 51% is achieved on the $\mathcal{M}$atchSet in both phases, while the $\mathcal{D}$odgeSet demonstrates a coverage of approximately 27% in Phase 1 and 43% in Phase 2. These results are remarkable considering that we are able to match 1,275 faces with only 10 Attack Faces and dodge 1,425 faces out of a total of 2,500. For a comprehensive breakdown of the results, please refer to Table 5.1.

It is important to note that Phase 2 of the algorithm generally exhibits lower performance compared to Phase 1, particularly in the more challenging scenarios characterized by larger sizes of $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet, particularly in terms of the coverage of the $\mathcal{D}$odgeSet. This observation is supported by Figure 5.6, which examines the coverage results achieved with a fixed $\mathcal{M}$atchSet size of 500 while varying the size of the $\mathcal{D}$odgeSet between 10 and 2,500. The coverage of the $\mathcal{M}$atchSet in both phases is higher when the $\mathcal{D}$odgeSet size is smaller, and it decreases as the $\mathcal{D}$odgeSet size increases.

Table 5.1: The table displays average outcomes from 5 runs of One Face to Rule Them All Algorithm's Phases 1 and 2, across different $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet sizes, based on 10 clusters. These results correspond with those in Figure 5.5.

$\lvert\mathcal{M}\text{atchSet}\rvert$	$\lvert\mathcal{D}\text{odgeSet}\rvert$	Coverage			
		$\overline{\mathcal{M}\text{atchSet}}$	$\overline{\mathcal{D}\text{odgeSet}}$	$\mathcal{M}$atchSet	$\mathcal{D}$odgeSet
10	0	100.00	0.00	100.00	0.00
10	1	100.00	0.00	100.00	0.00
10	2	100.00	0.00	100.00	0.00
10	3	100.00	0.00	100.00	0.00
10	10	100.00	0.00	100.00	10.00
10	100	100.00	0.00	100.00	3.25
10	500	100.00	0.00	100.00	4.15
10	1000	100.00	0.00	100.00	3.48
10	2500	100.00	0.45	100.00	2.48
100	0	77.75	0.00	74.00	0.00
100	1	77.00	0.00	73.50	0.00
100	2	77.50	0.00	74.50	0.00
100	3	77.75	0.00	74.50	8.33
100	10	77.25	0.00	75.25	22.50
100	100	76.25	0.00	74.25	13.25
100	500	73.75	0.05	70.25	9.35
100	1000	73.00	0.05	69.50	8.42
100	2500	66.33	0.13	65.33	5.16
500	0	68.85	0.00	65.15	0.00
500	1	66.50	0.00	62.55	0.00
500	2	67.40	0.00	63.35	12.50
500	3	67.70	0.00	64.10	16.67
500	10	67.55	5.00	63.50	20.00
500	100	65.65	3.50	62.47	24.00
500	500	54.40	1.15	52.35	18.55
500	1000	47.60	0.88	47.40	20.38
500	2500	39.20	0.45	41.40	16.67
1000	0	67.62	0.00	63.82	0.00
1000	1	65.42	0.00	61.52	0.00
1000	2	66.12	0.00	61.78	12.50
1000	3	66.45	0.00	62.68	8.33
1000	10	66.85	2.50	62.38	17.50
1000	100	64.47	11.33	60.70	33.67
1000	500	56.00	7.65	54.68	30.20
1000	1000	47.00	5.00	47.30	29.50
1000	2500	35.63	3.15	40.53	26.15
2500	0	65.86	0.00	61.56	0.00
2500	1	64.23	0.00	59.57	0.00
2500	2	64.88	0.00	61.31	12.50
2500	3	65.11	0.00	60.92	25.00
2500	10	65.16	12.50	61.01	50.00
2500	100	64.61	30.67	60.36	46.67
2500	500	61.47	29.75	58.34	41.85
2500	1000	59.25	30.20	57.29	43.60
2500	2500	51.42	27.24	51.20	43.40

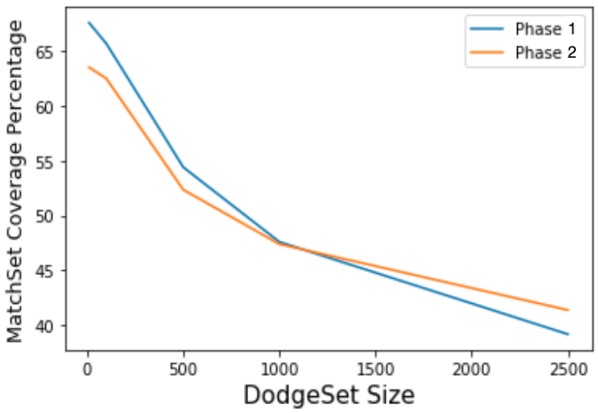

Figure 5.6: The coverage outcomes are reported for a fixed $\mathcal{M}$atchSet comprising 500 faces while considering various sizes for the $\mathcal{D}$odgeSet.

5.6.2 Multi Identity Impersonation or Master Face

5.6.2.1 Comparing the One Face to Rule Them All algorithm to other competing methods

In an extreme scenario known as Master Face Attack, which falls under Multi Identity Impersonation, the objective is to impersonate a large number of identities while having an empty $\mathcal{D}$odgeSet. Recently, Shmelkin et al. [121] proposed a method for the Master Face attack that involved generating a set of nine images using pre-trained GANs. This method achieved coverage of 43.82% across the 5,749 identities in the LFW dataset. However, the generated Master Faces did not have control over the specific identities. To compare our approach with this attack, we replicated the exact testing setup used in their work. We used FaceNet trained on CASIA-WebFace with a decision threshold corresponding to a false acceptance rate of 0.001, and we considered the 5,749 identities from the LFW dataset as the $\mathcal{M}$atchSet. We selected an image of Albert Einstein as the Source Face for our experiment, although any other Source Face could have been chosen. To ensure a fair comparison, we generated an equal number of images as Shmelkin et al. [121]. Figure 5.7 displays the Attack Faces generated using our One Face to Rule Them All method. The original Source Face is shown within a blue square, followed by the nine Attack Faces produced. Our proposed method significantly improves the achieved coverage while providing full control over the identities of the generated images. Using the set of nine images provided, we were able to attain a coverage rate of 58.5% for the first Einstein image (top) and 57.27% coverage for the second Einstein image (bottom). In comparison, the compet-

ing method achieved a coverage of only 43.82% with the same number of images, without any ability to control the identities. This demonstrates that our method can utilize any preferred Source Face to generate Attack Faces with superior coverage. The generated Attack Faces have modifications that are nearly imperceptible to humans, making the attack feasible without raising suspicion. Additionally, the results indicate that the nine visually identical images cover distinct regions of the face space, aligning with different identities.

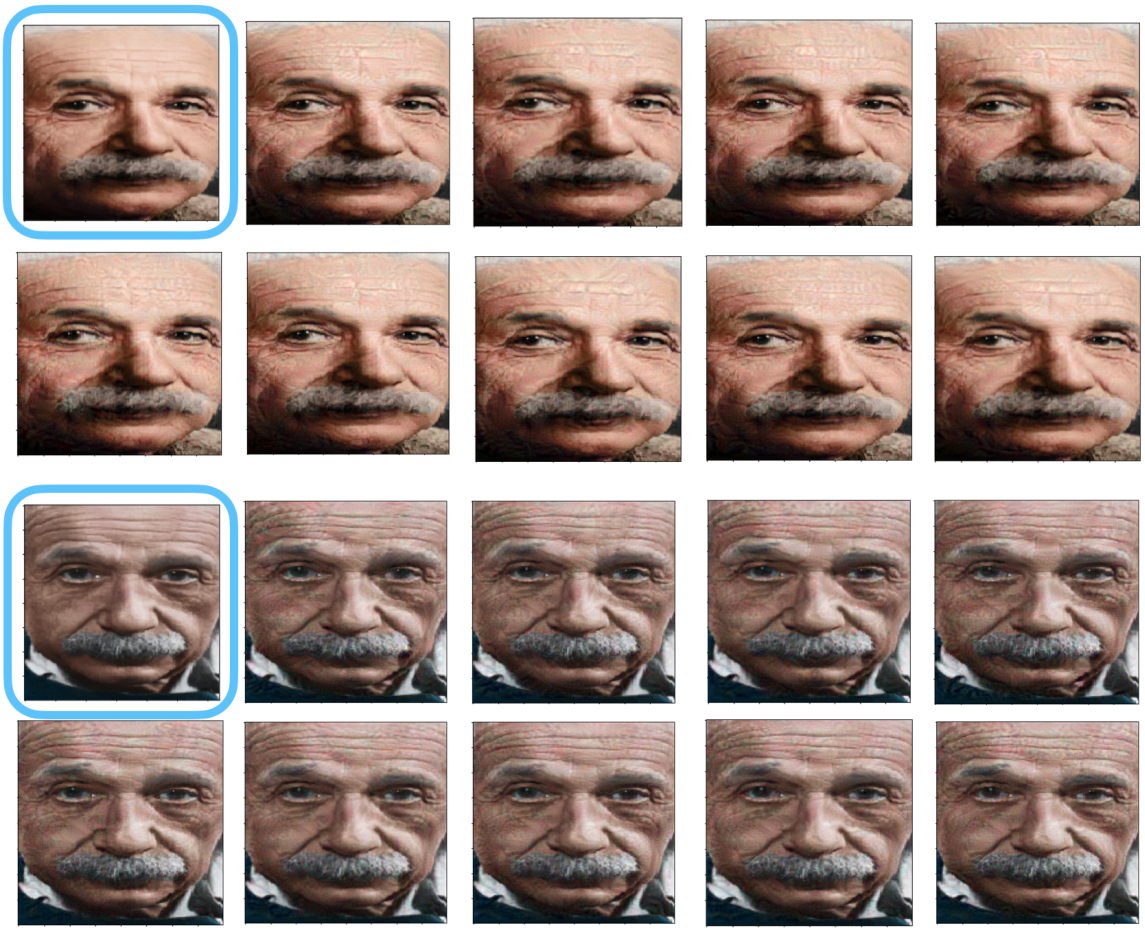

Figure 5.7: The One Face to Rule Them All Algorithm was used for Multi Identity Impersonation of 5,749 identities with two Albert Einstein images. The original image (Source Face) is in a blue box; the generated Attack Faces achieved 58.5% (top) and 57.27% (bottom) coverage rates, surpassing the previous method's 43.82%.

5.6.2.2 An analysis of the coverage achieved by the One Face to Rule Them All algorithm on Master Faces.

We conducted additional experiments to investigate the relationship between the number of Attack Faces generated and the coverage achieved on the $\mathcal{M}$atchSet, aiming to surpass a 90% coverage threshold. The findings of this investigation are presented in Figure 5.8. Notably, as the number of clusters and corresponding Attack Faces increased, the coverage rates also rose. Surprisingly, with a mere 80 Attack Face images, we were able to achieve a coverage exceeding 90%. This experiment underscores the fact that even with a relatively small number of images, we can effectively cover a significant portion of the $\mathcal{M}$atchSet. This observation suggests that the FaceNet model may cluster similar images together in the embedding space. Importantly, our attack demonstrates the ability to compromise FV systems using a limited number of Attack Faces, thereby exposing the vulnerability and fragility of these systems. The fact that a mere 80 Attack Faces resulted in a 90% match rate underscores the insecurity of FV systems, as they can be breached with a relatively small number of attempts.

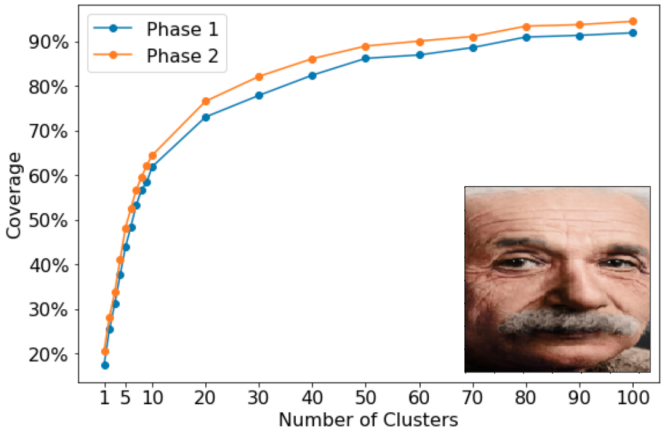

Figure 5.8: The plot illustrates the coverage achieved for various numbers of clusters (or Attack Faces) on a $\mathcal{M}$atchSet that includes 5,749 identities from the LFW dataset. The Source Face utilized in the experiment is displayed in the plot.

5.6.2.3 On the Generalizability of One Face to Rule Them All Attack on Master Faces

The purpose of this experiment is to evaluate the applicability of the Attack Faces generated using the One Face to Rule Them All method to unseen identities in the Master Face Attack scenario. In order to assess this, we examine how well the Attack Faces cover identities that were not part of the generation process. We construct a $\mathcal{M}$atchSet consisting of a randomly selected set of 2,500 identities and generate a set of 10 Attack Faces to impersonate these identities. Additionally, we create a separate set of 2,500 identities called the $\mathcal{U}$nseenSet, which exclusively includes identities not present in the $\mathcal{M}$atchSet. Finally, we measure the coverage achieved by the 10 Attack Faces on both the $\mathcal{M}$atchSet and the $\mathcal{U}$nseenSet. This test is repeated five times, and the average results are reported.

The outcomes of this study reveal that by employing 10 Attack Faces, we attain a coverage rate of 65.78% and 60.22% for the $\mathcal{M}$atchSet in Phase 1 and Phase 2, respectively. **In addition, the identical set of 10 Attack Faces covers 58.25% and 56.9% of the members in the $\mathcal{U}$nseenSet in Phase 1 and Phase 2, respectively.** These results provide compelling evidence of the generalizability capacity inherent in our proposed One Face to Rule Them All method.

5.6.3 Single Impersonation and Single Dodging Scenario

Single Impersonation Scenario. In this study, a single picture was randomly selected from $\mathcal{F}$ to be the $\mathcal{M}$atchSet, accompanied by an empty $\mathcal{D}$odgeSet. The goal was to alter Albert Einstein's face image to bear a resemblance to the identity portrayed in the chosen picture. The experiment was repeated ten times, and the results consistently demonstrated a 100% coverage rate. This indicates that in all ten trials, the modified image of Einstein successfully impersonated the randomly selected individual.

Single Dodging Scenario. To tackle the single dodging scenario, an image is randomly chosen from $\mathcal{F}$ to serve as the $\mathcal{D}$odgeSet, while an empty $\mathcal{M}$atchSet is established. The selected image is then modified to evade or avoid its associated identity. Consequently, this image becomes the sole member of the $\mathcal{D}$odgeSet. The experiment is replicated ten times, and it is observed that in all ten instances, the dodging objective is accomplished successfully (0% coverage).

In the context of a FV system that restricts the number of permissible attempts, an attacker's capability to infiltrate becomes more complex. Nevertheless, given that these two scenarios yielded a 100% success rate, our strategy has the potential to circumvent

such systems even with limited tries. This holds generally true for cases of impersonating a limited set of individuals (referring to Section 5.6.1). The challenge intensifies in more complex scenarios, such as a Master Face Attack.

5.6.4 Fitness Function Ablation Studies

5.6.4.1 Sensitivity of $\mathcal{D}$odgeSet threshold

The fitness function utilized in our proposed GA for DodgePersonation (see Definition 5.5) incorporates two decision thresholds: $th1$ for $\mathcal{M}$atchSet and $th2$ for $\mathcal{D}$odgeSet. For our experimental setup, both $th1$ and $th2$ are fixed at a value of 1.055, as detailed in Section 5.5.3. In this particular investigation, we aim to examine the impact of varying $th2$ on the coverage achieved for both $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet.

In our proposed GA's DodgePersonation Fitness function (cf. Definition 5.5), we consider two decision thresholds: $th1$ for $\mathcal{M}$atchSet and $th2$ for $\mathcal{D}$odgeSet. For our experimental setup, both $th1$ and $th2$ are initially set to 1.055, as outlined in Section 5.5.3. The purpose of this experiment is to explore the impact of varying $th2$ on the coverage achieved by $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet. Our objective is twofold: (i) to assess if we can generate Attack Faces that are more effective at evading detection in $\mathcal{D}$odgeSet, and (ii) to understand the influence of adjusting $th2$ on the coverage of $\mathcal{M}$atchSet. We hypothesize that by increasing $th2$ and widening the margin of the $\overline{\mathcal{D}\text{odgeSet}}$, more points can be successfully dodged, as optimal points will be further away from the members of this set. We conducted multiple experiments, testing the initial $th2$ value of 1.055 as well as four additional $th2$ values representing a 3%, 4%, 5%, and 6% increase. Each experiment was repeated five times, using a randomly selected $\mathcal{M}$atchSet of 1,000 identities and a $\mathcal{D}$odgeSet of 500 identities. The average coverage results for Phase 1 and Phase 2 on $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet are presented in Table 5.2.

In Phase 1, the coverage of the $\mathcal{D}$odgeSet with the default $th2$ value of 1.055 is 7.65%, while in Phase 2, it is 30.20%. This means that our specialized GA was unable to evade 7.65% of the $\overline{\mathcal{D}\text{odgeSet}}$ members in the embedding space, and the generated Attack Faces failed to dodge 30.2% of the $\mathcal{D}$odgeSet cases.

When we increased the $\mathcal{D}$odgeSet threshold $th2$ by 3%, there was a significant decrease in the coverage percentages of the $\mathcal{D}$odgeSet in both phases. This confirms that raising the threshold $th2$ helps to keep the Attack Faces further away from the $\mathcal{D}$odgeSet cases. However, we noticed that this change had a negative impact on the coverage of the $\mathcal{M}$atchSet, resulting in decreased coverage.

Table 5.2: Coverage outcomes for distinct $th2$ values on Phase 1 and Phase 2 (Phase 2 values in parentheses).

$th2$	increase %	Coverage	
		$\mathcal{M}$atchSet	$\mathcal{D}$odgeSet
1.055	-	56.00 (54.68)	7.65 (30.20)
1.086	3%	41.56 (42.30)	0.00 (3.04)
1.097	4%	37.98 (39.50)	0.00 (1.48)
1.107	5%	30.46 (35.62)	0.00 (0.60)
1.118	6%	31.02 (33.72)	0.00 (0.40)

As we continued to increase the $th2$ threshold, the coverage of the $\mathcal{M}$atchSet continued to decline, while the coverage of the $\mathcal{D}$odgeSet approached zero. When we reached a 4% increase in $th2$, only the coverage of the $\mathcal{M}$atchSet was affected, as the coverage of the $\mathcal{D}$odgeSet was already very close to the desired value of zero.

Therefore, we can confirm that adjusting the threshold ratio for the $\mathcal{D}$odgeSet can lead to improved dodging results while sacrificing impersonation results. This trade-off should be carefully considered based on the nature of the problem at hand and the relative importance of dodging versus impersonation.

5.6.4.2 Sensitivity of parameter γ

The parameter γ in the GA Fitness function determines the weighting of the $DPloss$ for the $\overline{\mathcal{M}\text{atchSet}}$ and $\overline{\mathcal{D}\text{odgeSet}}$. We conducted experiments where we randomly selected 1,000 members from the $\overline{\mathcal{M}\text{atchSet}}$ and 500 members from the $\overline{\mathcal{D}\text{odgeSet}}$, and tested different values of γ including 0, 0.1, 0.3, 0.5, 0.7, 0.9, and 1. These experiments were repeated five times, and the average results were recorded. The findings of these experiments are depicted in Figure 5.9.

We observe distinct behavior based on the value of γ. When γ is set to 0, all the focus is on evading members of $\overline{\mathcal{D}\text{odgeSet}}$, leading to a neglect of $\overline{\mathcal{M}\text{atchSet}}$ members. Conversely, when γ is 1, there is a peak in the coverage of $\overline{\mathcal{M}\text{atchSet}}$ members, but no attention is given to dodging $\overline{\mathcal{D}\text{odgeSet}}$ members, resulting in a significant coverage of them. A value of 0.1 for γ yields considerable coverage for $\overline{\mathcal{M}\text{atchSet}}$ members, with only a minor increase in the coverage of $\overline{\mathcal{D}\text{odgeSet}}$. As γ continues to increase, the emphasis on $\overline{\mathcal{M}\text{atchSet}}$ grows, leading to increased coverage, while the focus on $\overline{\mathcal{D}\text{odgeSet}}$ diminishes, resulting in fewer

dodged members. Generally, the results remain relatively stable for values of γ ranging from 0.1 to 0.9.

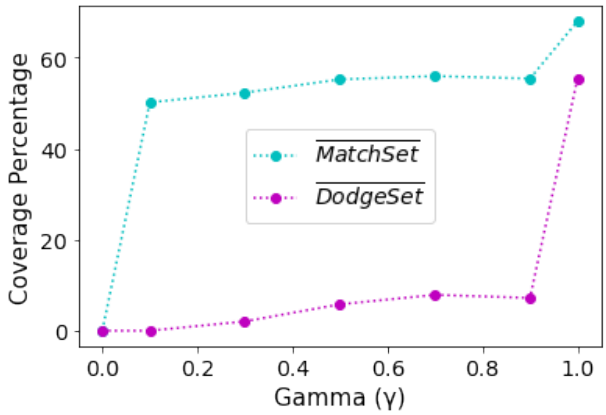

Figure 5.9: Impact of γ on the coverage of $\overline{\mathcal{M}\text{atchSet}}$ and $\overline{\mathcal{D}\text{odgeSet}}$.

5.6.5 Discussion on Using the $\mathcal{D}$odgeSet in the DodgePersonation Attack

The proposed definition of DodgePersonation Attack needs the presence of both a $\mathcal{M}$atchSet and a $\mathcal{D}$odgeSet. One could argue that including a $\mathcal{D}$odgeSet might be redundant since matching the identity or identities in the $\mathcal{M}$atchSet should inherently lead to the dodging of any other images. However, we contend that this assumption may not always hold true, highlighting that using both the $\mathcal{M}$atchSet and the $\mathcal{D}$odgeSet is essential.

To illustrate this point, let us consider an example. Suppose we are faced with the task of creating an image that appears as person A to human observers, is recognized by the FV system as person B, and successfully avoids being identified as person C (or dodges person C). While some may argue that if the FV system confirms the image as person B, it automatically avoids other identities like person C, this assumption is not guaranteed. FV systems exhibit imperfections, as evidenced by the existence of Master Faces, meaning that a solution may match more identities than those explicitly specified in the $\mathcal{M}$atchSet. To address this concern and ensure the system's robustness and reliability, the inclusion of a $\mathcal{D}$odgeSet becomes necessary, explicitly taking into account the unwanted identities.

5.6.6 Exploring the Identities Distribution in the Embedding Space

To gain a deeper understanding of the distribution of face images from different identities in the embedding space, two experiments were conducted. The first experiment involved randomly selecting an image as the $\mathcal{M}$atchSet, leaving the $\mathcal{D}$odgeSet empty, and modifying Albert Einstein's face image to impersonate it. This process was repeated five times, and the average result was recorded. Following that, the same process was repeated with an increasing number of images in the $\mathcal{M}$atchSet, i.e., using two images representing different identities, then three images of distinct identities, and so forth, until reaching one hundred images corresponding to one hundred unique identities. Each experiment using a given number of distinct identities was repeated five times. The results, shown in Figure 5.10, clearly demonstrate that as the number of identities that we aim to impersonate increases, the coverage percentage decreases. This observation suggests that the embeddings of these identities are not concentrated in a specific region of the embedding space, making it impossible for a single point to accurately represent all of them.

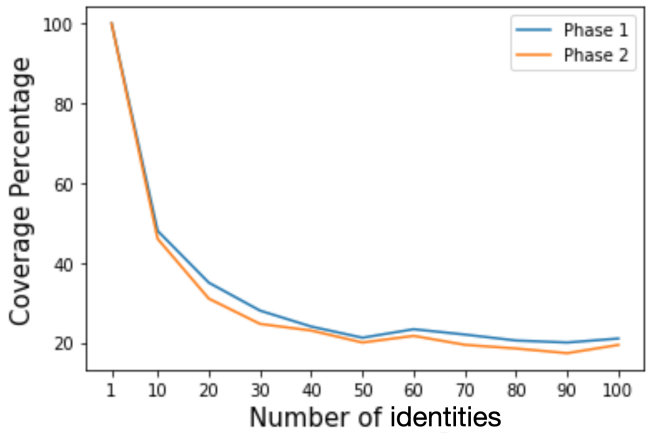

Figure 5.10: Percentage of coverage based on the number of identities to be impersonated using a single Attack Face image.

5.6.7 Study of Alternative in Phase 2: Projected Mean

It is possible that the average of all the $\overline{\mathcal{M}\text{atchSet}_i}$ members is close enough to most of its members. However, there is no guarantee that it is far enough from the $\overline{\mathcal{D}\text{odgeSet}}$ members. Our experiments assess whether the arithmetic mean of an embedding set is closer to all its members than the decision threshold. For the FM that we currently use, a

Table 5.3: The results of projected mean and GA search for $\mathcal{M}$atchSet of size 100 and $\mathcal{D}$odgeSet size of size 0, 100, 500. The results are the average of 5 runs for 10 clusters.

$\lvert\mathcal{M}\text{atchSet}\rvert$	$\lvert\mathcal{D}\text{odgeSet}\rvert$	method (Phase 2)	$\mathcal{M}$atchSet coverage(%)	$\mathcal{D}$odgeSet coverage(%)
100	0	GA	77.75	0.00
100	0	Projected Mean	76.50	0.00
100	100	GA	76.25	0.00
100	100	Projected Mean	76.50	29.25
100	500	GA	73.75	0.05
100	500	Projected Mean	76.50	29.60

slight adjustment is necessary after the arithmetic mean computation. The FM we employ normalizes the extracted features using L2-normalization, resulting in unit length vectors. Hence, our embedding space only occupies a fraction of all potential vectors within that multi-dimensional space. Specifically, because of L2-normalization, our FM confines itself to the surface of a hyper-sphere in this multi-dimensional domain. Consequently, the average point derived from separate points on this hyper-sphere's surface will be within the hyper-sphere. To address this, we project this average point back to the hyper-sphere's surface.

Table 5.3 presents the outcomes from multiple experiments that utilized both the projected mean technique and the GA search method across varying sizes of $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet. The results indicate that the projected mean method yields good candidate points when $\overline{\mathcal{D}\text{odgeSet}}$ is empty. The contrast is more clear in difficult cases where $\overline{\mathcal{D}\text{odgeSet}}$ is not empty. Therefore, the GA works much better than a simple projection of the average of $\overline{\mathcal{M}\text{atchSet}}$ onto the surface of the unit hyper-sphere.

5.6.8 Generalizability of Attack Faces

We performed an experiment to investigate the generalizability of the generated Attack Faces. We tested whether a generated attacked image is able to match different images of the same identity in the $\mathcal{M}$atchSet and dodge different images of the same identity in the $\mathcal{D}$odgeSet. This means that we aim to assess the coverage of an Attack Face on images that are not included in these sets but are images from the same identities that are in these sets.

We randomly selected one identity for the $\mathcal{M}$atchSet and one identity for the $\mathcal{D}$odgeSet. Then, for each identity, we chose 10 images, only one of which was shown to the DodgePer-

sonation Attack during training. The remaining nine images were kept isolated until the Attack Faces were crafted. We then tested the crafted Attack Faces on both the image in the $\mathcal{M}$atchSet and $\mathcal{D}$odgeSet, and the 9 images that were kept isolated. The experiments were repeated 5 times and we observed the average coverage results.

The results show that in phase 2, there was 100% coverage for the one image in the $\mathcal{M}$atchSet. The subsequent nine images, which are different images of the same identity and were not exposed to the algorithm during training (as they are not in the $\mathcal{M}$atchSet), had a coverage of 73.33%. In contrast, for the $\mathcal{D}$odgeSet images, the coverage was 0% for both the image included in the $\mathcal{D}$odgeSet and the remaining nine images. These findings suggest that the Attack Faces can be generalized to other previously unseen images of the same identity. However, to validate this finding further and establish the generalizability of the Attack Faces, more extensive tests are required.

5.6.9 Study of Two Extreme Cases

This section explores two particular extreme cases. In the first one, we generate an image using the most distant embedding of another image, and in the second one, we observe the impact of using a black image as the source image to obtain an attack image. We describe these two cases in more detail next.

Generating an Image with an Opposite Embedding

Our starting point is an existing image, from which we obtain the corresponding embedding. We will generate an attack image based on an embedding that is the farthest away from the embedding of this image. Our goal is to observe the resulting attack image in this scenario.

The Embedding Space of our FM (FaceNet) consists of 512 dimensions, and due to the normalization of vectors, each vector has a unit radius. Consequently, the Embedding Space can be thought of as the surface of a 512-dimensional hyper-sphere with a unit radius. Since every point in this space is one unit away from the origin in Euclidean distance, the maximum distance between any two points is two units.

For our initial image, we selected one Einstein image that was used in our previous experiments (top image in Figure 5.11). We then obtained the embedding corresponding to this image ($\overline{Einstein} = FM(Einstein)$). By negating each element of $\overline{Einstein}$, we obtained $\overline{Einstein_p}$ such that the distance between $\overline{Einstein}$ and $\overline{Einstein_p}$ is exactly 2 units ($Dist(\overline{Einstein}, \overline{Einstein_p}) = 2$). Finally, we used the initially selected Einstein image as the Source Face and applied the Attack Face Generation method to obtain an

image corresponding to the given embedding, and that resembles the original image. We generated three images, each using a different value for the perturbation parameter of our algorithm. We selected the following perturbation values for this experiment 0.15, 0.075, and 0.0375.

The results obtained are shown in Figure 5.11. The top image is the original image selected and the three bottom images correspond to the three Attack Faces obtained (less perturbation on the left, more perturbation on the right). We observe that the most perturbed image has a distance of 1.998 from the original, while the least perturbed had a distance of 1.860. Despite their visual similarity to the selected Einstein image, the generated images are nearly the most distant identities according to FaceNet, as the farthest point from a given point in the embedding space is 2 units.

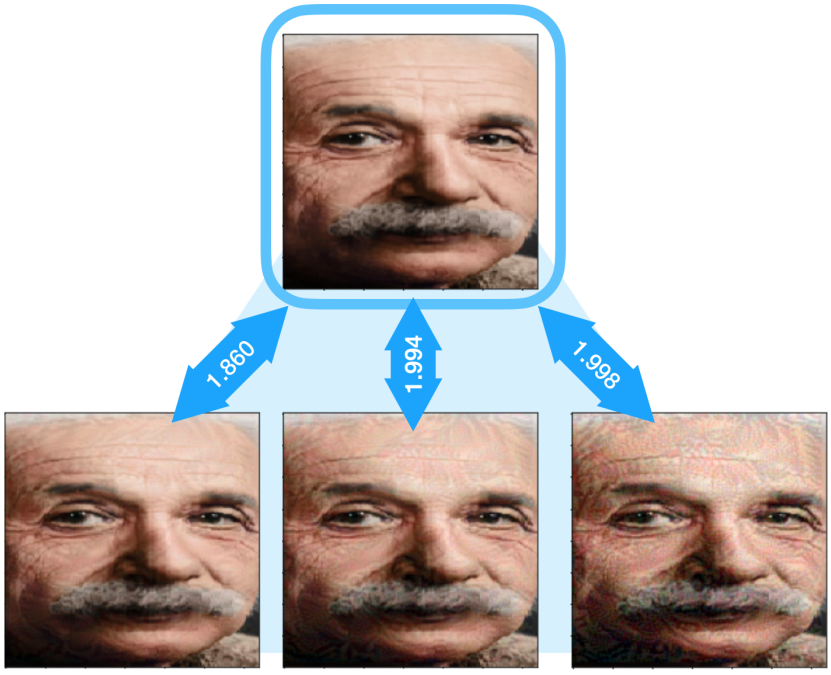

Figure 5.11: Results of three experiments with varying perturbation degrees. Original Einstein image on the top; Attack Faces on the bottom with perturbation degrees of 0.0375 (left), 0.075 (middle), and 0.15 (right).

Generating an Attack Face from a Black Image

In this study, our goal is to observe the impact of using a black image as a source face to generate an attack face. We aim at understanding how close the attack face generated

is to the target and how it looks visually. Moreover, we also compare the generated attack with other images of the same identity.

Following our previous experiments we selected two Einstein images (right and left images in Figure 5.12). We start with one of these images, which we will represent as $Einstein$ (that is depicted on the left side of Figure 5.12), and deployed the Attack Face Generation method to adjust a black image ($Black_{adv}$) such that $FM(Black_{adv}) \approx \overline{Einstein}$, i.e., the attack image generated will match the embedding of the Einstein image.

In the following sequence of experiments, we began by acquiring the embedding for the Einstein image (depicted in the blue box on the left side of Figure 5.12), denoted as $\overline{Einstein} = FM(Einstein)$. We then utilized a completely black image as the Source Face image and employed the Attack Face Generation method to adjust the Source Face such that $FM(Black_{adv}) \approx \overline{Einstein}$. The objective was to explore the visual outcome when manipulating the black image to closely resemble Einstein from the perspective of FaceNet. Figure 5.12 illustrates the distances between the manipulated black image at the top, the unaltered Einstein images in the middle, and the modified Einstein image from the previous experiment with a 0.075 perturbation degree at the bottom. Despite the generated image and Einstein's image being considered highly similar in the embedding space, there is no noticeable resemblance between them when perceived by the human eye in the Image space.

Notably, it is worth recalling that the decision threshold for a false acceptance rate of 0.001 is 1.055. In Figure 5.12, we observed that the Euclidean distance between the embeddings of the unchanged Einstein image on the left and the unchanged Einstein image on the right is 0.660, significantly lower than the decision threshold. Hence, FaceNet identifies these two images as belonging to the same identity, with their distance accounting for only 33% of the maximum distance between any two points in the embedding space. Considering this, the results demonstrate that the Euclidean distance between the left Einstein image and the modified black image in Figure 5.12 is a mere 0.046. This distance is not only small enough for FaceNet to consider the images to belong to the same identity, but it is also merely 2.3% of the maximum possible distance between any two points in the embedding space.

5.7 Ethical Considerations

Should malevolent actors exploit the proposed attack, systems that employ FV may be compromised. Nevertheless, it is important to disclose the inherent vulnerabilities of such

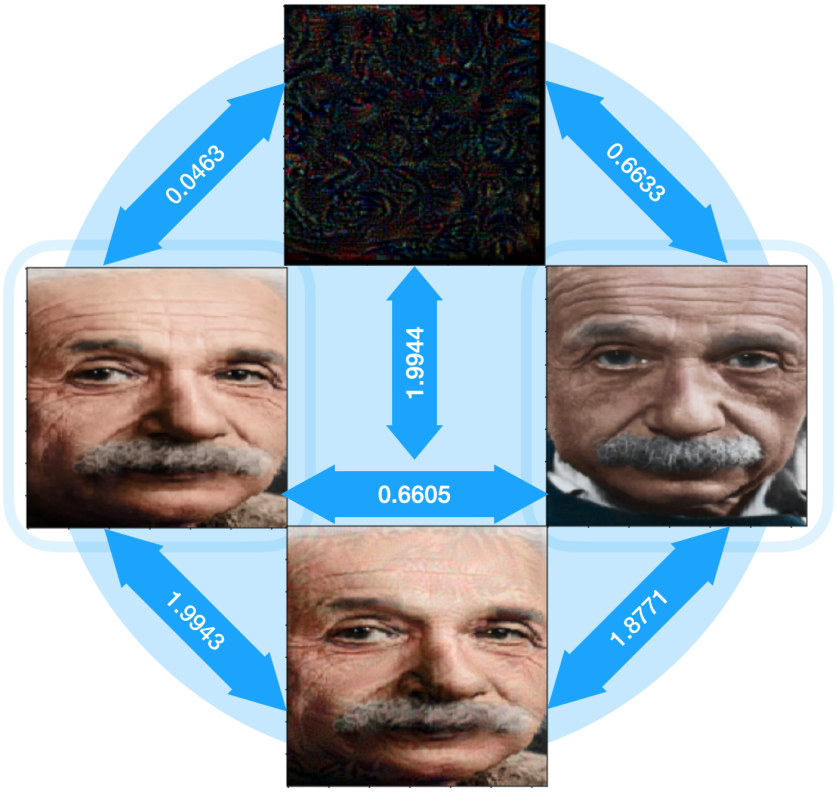

Figure 5.12: On the visual observations of the attacked images.

systems to improve security. Through our research, we aim to clarify some aspects of the frailties of FV systems and advocate for the academic community to devise protective strategies or methodologies against these identified vulnerabilities. We posit that the beneficial outcomes of our research will supersede the potential risks. Moreover, it is our contention that, had malicious individuals discovered this vulnerability, they would likely have withheld such information, thereby leaving systems vulnerable without allowing the research community the opportunity to understand the nature of the threat and develop appropriate countermeasures. Finally, the proposed attack could potentially be a basis for defence mechanism that not only diminishes this type of attack but also could potentially improve the accuracy of the FV system. Lastly, the suggested attack could serve as a foundation for developing defensive measures that not only counteract this type of attack but also possibly enhance the system's accuracy.

5.8 Conclusion

DodgePersonation Attack was introduced as a method to target FV systems, encompassing a comprehensive definition and taxonomy that covers both existing and innovative attack types. This attack aims to generate images that can impersonate members of the $\mathcal{M}$atchSet while evading detection by the $\mathcal{D}$odgeSet members. A novel approach named "One Face to Rule Them All" was proposed, effectively deploying the DodgePersonation Attack by generating adversarial images capable of impersonating a specific set of identities while avoiding the identities from another set. The proposed algorithm achieves exceptional performance in known attack types, surpassing current benchmarks, and demonstrates promising outcomes in novel attack scenarios. Compared to a previous study addressing the Master Face Attack, which achieved coverage of 43.82% of the dataset using nine images without any identity constraints, our method significantly improves coverage to over 57% with the same number of images with an identity constraint. Lastly, it is important to emphasize that the generated Attack Faces are meticulously crafted to introduce minimal changes that can go unnoticed by the human eye.

5.9 Future Works

In terms of future work, one direction to explore is the generalizability of our solution to different FMs. This entails examining how robust the solution is to variations in the feature extraction process, which could affect the system's reliability in real-world scenarios. It

is also important to investigate the generalizability of our method across different images of the same identity. The application of our method to 3D images should be considered, aligning with the emerging trend in related research [31]. Additionally, it would be valuable to experiment with the printed version of Attack Faces for conducting Physical or Presentation Attacks. Exploring the potential of Adversarial Patches is also promising. These involve altering a smaller region of the Source Face with a more intense level of distortion to create a patch. This patch can then be applied to a face image. This approach is applicable to both Digital Attacks, as discussed in this chapter, and Physical Attacks. Lastly, extending the proposed algorithm to fingerprint recognition systems and investigating its effectiveness in generating a master fingerprint set can also be considered.

Chapter 6

Conclusion

This research has made several contributions to the field of DL and Cybersecurity. Initially, we use GANs as a means of oversampling to tackle the problem of class imbalance. This issue is particularly relevant for predictive tasks in Cybersecurity. Next, we introduced an innovative training strategy for GANs, enabling their practical application on datasets that do not allow a visual inspection of the outputs. This characteristic is shared by many Cybersecurity datasets, which consist of tabular, non-imagery data. Lastly, we directed our attention towards examining the resilience of DL-based FV systems and highlighted new concerns regarding their vulnerability in the presence of an adversary.

6.1 Summary of Contributions

In Chapter 3, the suitability of CGAN as an oversampling strategy was studied in the context of predictive tasks involving multiple data difficulty factors. The impact of various factors such as class overlap, data dimensionality, imbalance ratio, and sample size was examined in imagery datasets, as well as the challenges posed by class imbalance and small sample size in Cybersecurity datasets. The experimental results demonstrated that CGAN-based oversampling effectively addressed these data difficulty factors, resulting in improved performance, particularly for the minority class. The study concluded that CGANs are effective in handling multiple data difficulty factors encountered in real-world problems.

In Chapter 4, a solution called AutoGAN was proposed to automate the process of determining when to stop training a GAN. Through extensive experiments on tabular and imagery data, AutoGAN achieved comparable or better results than other methods,

while requiring fewer training iterations. The performance of the minority class improved or remained comparable, while the majority class was largely unaffected. The findings demonstrated that AutoGAN can reduce the time associated with manual human inspection during GAN training for imagery datasets and also pave the way for GAN training on non-imagery datasets.

In Chapter 5, DodgePersonation Attack was introduced as a method to target FV systems. A novel approach named "One Face to Rule Them All" was proposed, which effectively deployed DodgePersonation Attack by generating adversarial images capable of impersonating specific identities while evading detection by other pre-determined identities. The algorithm achieved exceptional performance in known attack types and demonstrated promising outcomes in novel attack scenarios. Particularly noteworthy is the substantial improvement in coverage achieved by our method compared to a previous study that dealt with a known attack type called the Master Face Attack. While the prior study managed to cover 43.82% of the dataset using nine images without any identity constraints, our method significantly enhanced coverage to over 57% with the same number of images while imposing identity constraints.

6.2 Limitations and Future Works

The findings of this study offer important perspectives on the issue of class imbalance using GANs, the complexity of training a GAN, and a vulnerability within FV. However, several limitations need to be recognized. Firstly, as described in Chapter 3, a GAN is a complex model with hundreds of thousands to millions of adjustable parameters. Training a GAN for class imbalance is challenging, whereas other methods like SMOTE do not present these difficulties. Secondly, in Chapter 4, we do not provide a guarantee for successful GAN training. Moreover, our method does not reveal if the training process is affected by known problems like mode collapse. Thirdly, in Chapter 5, our tests were limited to attacking FV in the digital sphere, neglecting the real-world factor where a physical element, such as a camera, is involved. These limitations present opportunities for future research.

Regarding future works, we can highlight several avenues. In Chapter 3 the impact of using GANs when dealing with other data difficulty factors such as class overlap can be further explored. For the AutoGAN solution developed in Chapter 4, one promising path may involve adjusting the framework to detect issues like mode collapse. Continuing from Chapter 5, the practical application of the "One Face to Rule Them All" algorithm

in physical attacks should be examined. A more exhaustive list of possible future research paths can be found in each relevant chapter of this thesis.

References

[1] Daniel González Alé, N. R. Wilfred Blessing, and Ogunti Erastus. Object recognition using deep learning. *International journal of engineering research and technology*, 7, 2018.

[2] Takuma Amada, Seng Pei Liew, Kazuya Kakizaki, and Toshinori Araki. Universal adversarial spoofing attacks against face recognition. In *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–7. IEEE, 2021.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[4] Kourosh T. Baghaei, Amirreza Payandeh, Pooya Fayyazsanavi, Shahram Rahimi, Zhiqian Chen, and Somayeh Bakhtiari Ramezani. Deep representation learning: Fundamentals, perspectives, applications, and open challenges, 2022.

[5] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, page 26–33, USA, 2001. Association for Computational Linguistics.

[6] Shane Barratt and Rishi Sharma. A note on the inception score, 2018.

[7] Samyadeep Basu, Rauf Izmailov, and Chris Mesterharm. Membership model inversion attacks for deep networks, 2019.

[8] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International conference on machine learning*, pages 610–619. PMLR, 2018.

[9] Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[10] Ali Borji. Pros and cons of gan evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329, 2022.

[11] Paula Branco and Luis Torgo. A study on the impact of data characteristics in imbalanced regression tasks. In *2019 IEEE DSAA*, pages 193–202. IEEE, 2019.

[12] Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):1–50, 2016.

[13] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018.

[14] Dariusz Brzezinski, Leandro L Minku, Tomasz Pewinski, Jerzy Stefanowski, and Artur Szumaczuk. The impact of data difficulty factors on classification of imbalanced and concept drifting data streams. *KAIS*, 63(6):1429–1469, 2021.

[15] Jie Cao, Yibo Hu, Bing Yu, Ran He, and Zhenan Sun. 3d aided duet gans for multi-view face image synthesis. *IEEE Transactions on Information Forensics and Security*, 14(8):2028–2042, 2019.

[16] Efstathios Chatzikyriakidis, Christos Papaioannidis, and Ioannis Pitas. Adversarial face de-identification. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 684–688, 2019.

[17] Poonam Chaudhari, Himanshu Agrawal, and Ketan Kotecha. Data augmentation using mg-gan for improved cancer classification on gene expression data. *Soft Computing*, 24(15):11381–11391, Aug 2020.

[18] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[19] Jiawei Chen, Janusz Konrad, and Prakash Ishwar. Vgan-based image representation learning for privacy-preserving facial expression recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1570–1579, 2018.

[20] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

[21] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism, 2017.

[22] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694, 2019.

[23] Jiankang Deng, Yuxiang Zhou, and Stefanos Zafeiriou. Marginal loss for deep face recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2006–2014, 2017.

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

[25] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017.

[26] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[27] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7714–7722, 2019.

[28] Hajar Emami, Majid Moradi Aliabadi, Ming Dong, and Ratna Babu Chinnam. Spagan: Spatial attention gan for image-to-image translation. *IEEE Transactions on Multimedia*, 23:391–401, 2021.

[29] Justin Engelmann and Stefan Lessmann. Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications*, 174:114582, 2021.

[30] Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 479:448–455, 2019.

[31] Tomer Friedlander, Ron Shmelkin, and Lior Wolf. Generating 2d and 3d master faces for dictionary attacks with a network-assisted latent space evolution. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, pages 1–1, 2022.

[32] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. Autogan-distiller: Searching to compress generative adversarial networks. *arXiv preprint arXiv:2006.08198*, 2020.

[33] Pedro J. García-Laencina, José-Luis Sancho-Gómez, and Aníbal R. Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, September 2009.

[34] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3224–3234, 2019.

[35] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks, 2017.

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

[37] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[38] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[39] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[40] Dou Goodman, Hao Xin, Wang Yang, Wu Yuesheng, Xiong Junfeng, and Zhang Huan. Advbox: a toolbox to generate adversarial examples that fool neural networks. *arXiv preprint arXiv:2001.05574*, 2020.

[41] Noah F. Greenwald, Geneva Miller, Erick Moen, Alex Kong, Adam Kagel, Thomas Dougherty, Christine Camacho Fullaway, Brianna J. McIntosh, Ke Xuan Leow, Morgan Sarah Schwartz, Cole Pavelchek, Sunny Cui, Isabella Camplisson, Omer Bar-Tal, Jaiveer Singh, Mara Fong, Gautam Chaudhry, Zion Abraham, Jackson Moseley, Shiri Warshawsky, Erin Soon, Shirley Greenbaum, Tyler Risom, Travis Hollmann, Sean C. Bendall, Leeat Keren, William Graf, Michael Angelo, and David Van Valen. Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. *Nature Biotechnology*, 40(4):555–565, November 2021.

[42] Hongjiao Guan, Yingtao Zhang, Min Xian, Heng-Da Cheng, and Xianglong Tang. Smote-wenn: Solving class imbalance and small sample problems by oversampling and distance scaling. *Applied Intelligence*, 51:1394–1409, 2021.

[43] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[44] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information, 2017.

[45] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks, 2018.

[46] Arash Habibi Lashkari., Gerard Draper Gil., Mohammad Saiful Islam Mamun., and Ali A. Ghorbani. Characterization of tor traffic using time based features. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP,*, pages 253–262. INSTICC, SciTePress, 2017.

[47] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[48] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[49] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks*, pages 1322–1328. IEEE, 2008.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[52] 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. 2017.

[53] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[54] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[55] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[56] Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *SIGKDD Explor. Newsl.*, 6(1):40–49, jun 2004.

[57] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

[58] Joakim Kargaard, Tom Drange, Ah-Lian Kor, Hissam Twafik, and Emlyn Butterfield. Defending it systems against intelligent malware. In *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pages 411–417, 2018.

[59] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[60] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks, 2021.

[61] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[62] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2019.

[63] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

[64] Bach Ngoc Kim, Jose Dolz, Pierre-Marc Jodoin, and Christian Desrosiers. Privacy-net: An adversarial approach for identity-obfuscated segmentation of medical images, 2020.

[65] Stepan Komkov and Aleksandr Petiushko. Advhat: Real-world adversarial attack on arcface face id system. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 819–826, 2021.

[66] Michał Koziarski, Colin Bellinger, and Michał Woźniak. RB-CCR: Radial-based combined cleaning and resampling algorithm for imbalanced data classification. *Machine Learning*, 110(11-12):3059–3093, October 2021.

[67] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[68] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[69] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[70] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[71] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2016.

[72] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. Generative adversarial trainer: Defense to adversarial perturbations with gan, 2017.

[73] Qinya Li, Zhenzhe Zheng, Fan Wu, and Guihai Chen. Generative adversarial networks-based privacy-preserving 3d reconstruction. In *2020 IEEE/ACM 28th International Symposium on Quality of Service*, pages 1–10, 2020.

[74] Guanxiong Liu, Issa Khalil, and Abdallah Khreishah. Gandef: A gan based adversarial training defense for neural network classifier. In *ICT Systems Security and Privacy Protection - 34th IFIP TC 11 International Conference, SEC 2019, Proceedings*, IFIP Advances in Information and Communication Technology, pages 19–32. Springer New York LLC, 2019.

[75] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[76] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[77] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141, 2013.

[78] Ilya Loshchilov, Tobias Glasmachers, and Hans-Georg Beyer. Large scale black-box optimization by limited-memory matrix adaptation. *IEEE Transactions on Evolutionary Computation*, 23(2):353–358, 2019.

[79] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on LFW with GaussianFace. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), March 2015.

[80] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on lfw with gaussianface. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Mar. 2015.

[81] Yonghong Luo, Xiangrui Cai, Ying ZHANG, Jun Xu, and Yuan xiaojie. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[82] Samaneh Mahdavifar, Andi Fitriah Abdul Kadir, Rasool Fatemi, Dima Alhadidi, and Ali A. Ghorbani. Dynamic android malware category classification using semi-supervised deep learning. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 515–522, 2020.

[83] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A. Ghorbani. Detecting malicious urls using lexical analysis. In *Network and System Security*, pages 467–482, Cham, 2016. Springer International Publishing.

[84] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

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

[86] Mohammadreza MontazeriShatoori, Logan Davidson, Gurdip Kaur, and Arash Habibi Lashkari. Detection of doh tunnels using time-series classification of encrypted traffic. In *2020 IEEE DASC/PiCom/CBDCom/CyberSciTech*, pages 63–70, 2020.

[87] Stanislav Morozov, Andrey Voynov, and Artem Babenko. On self-supervised image representations for {gan} evaluation. In *International Conference on Learning Representations*, 2021.

[88] Andres Munoz, Mohammadreza Zolfaghari, Max Argus, and Thomas Brox. Temporal shift gan for large scale video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3179–3188, 2021.

[89] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. Learning from imbalanced data in presence of noisy and borderline examples. In *International conference on rough sets and current trends in computing*, pages 158–167. Springer, 2010.

[90] Ehsan Nazari and Paula Branco. On oversampling via generative adversarial networks under different data difficulty factors. In *Proceedings of the Third International Workshop on Learning with Imbalanced Domains: Theory and Applications*, volume 154 of *Proceedings of Machine Learning Research*, pages 76–89. PMLR, 17 Sep 2021.

[91] Ehsan Nazari, Paula Branco, and Guy-Vincent Jourdan. Using cgan to deal with class imbalance and small sample size in cybersecurity problems. In *2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE, 2021.

[92] Ehsan Nazari, Paula Branco, and Guy-Vincent Jourdan. Autogan: An automated human-out-of-the-loop approach for training generative adversarial networks. *Mathematics*, 11(4), 2023.

[93] Huy H Nguyen, Sébastien Marcel, Junichi Yamagishi, and Isao Echizen. Master face attacks on face recognition systems. *arXiv preprint arXiv:2109.03398*, 2021.

[94] Huy H Nguyen, Junichi Yamagishi, Isao Echizen, and Sébastien Marcel. Generating master faces for use in performing wolf attacks on face recognition systems. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–10. IEEE, 2020.

[95] Artem Obukhov and Mikhail Krasnyanskiy. Quality assessment method for gan based on modified metrics inception score and fréchet inception distance. In *Software Engineering Perspectives in Intelligent Systems*, pages 102–114, Cham, 2020. Springer International Publishing.

[96] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 7:36322–36333, 2019.

[97] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[98] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proc. VLDB Endow.*, 11(10):1071–1083, June 2018.

[99] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, jun 2018.

[100] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016.

[101] Mikhail Pautov, Grigorii Melnikov, Edgar Kaziakhmedov, Klim Kireev, and Aleksandr Petiushko. On adversarial patches: Real-world attack on ArcFace-100 face recognition system. In *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*. IEEE, oct 2019.

[102] Matteo Pennisi, Simone Palazzo, and Concetto Spampinato. Self-improving classification performance through gan distillation. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1640–1648, 2021.

[103] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences*, 9(5), 2019.

[104] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[105] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. On wasserstein two sample testing and related families of nonparametric tests, 2015.

[106] Suman Ravuri and Oriol Vinyals. Classification accuracy score for conditional generative models, 2019.

[107] Arun Ross and Anil Jain. Information fusion in biometrics. *Pattern Recognition Letters*, 24(13):2115–2125, 2003. Audio- and Video-based Biometric Person Authentication (AVBPA 2001).

[108] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[109] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[110] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models, 2018.

[111] Vignesh Sampath, Iñaki Maurtua, Juan José Aguilar Martín, and Aitor Gutierrez. A survey on generative adversarial networks for imbalance problems in computer vision tasks. *Journal of Big Data*, 8(1), January 2021.

[112] Vignesh Sampath, Iñaki Maurtua, Juan José Aguilar Martín, and Aitor Gutierrez. A survey on generative adversarial networks for imbalance problems in computer vision tasks. *Journal of big Data*, 8(1):1–59, 2021.

[113] Miriam Seoane Santos, Pedro Henriques Abreu, Nathalie Japkowicz, Alberto Fernández, Carlos Soares, Szymon Wilk, and João Santos. On the joint-effect of class imbalance and overlap: A critical review. *Artif. Intell. Rev.*, 55(8):6207–6275, dec 2022.

[114] Divya Saxena and Jiannong Cao. D-gan: Deep generative adversarial nets for spatio-temporal prediction, 2019.

[115] Thorsten Schmidt. Coping with copulas. *Copulas-From theory to application in finance*, 3:1–34, 2007.

[116] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[117] Maryam Shahpasand, Len Hamey, Dinusha Vatsalan, and Minhui Xue. Adversarial attacks on mobile malware detection. In *2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile*, pages 17–20, 2019.

[118] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security (TOPS)*, 22(3):1–30, 2019.

[119] Shiwei Shen, Guoqing Jin, Ke Gao, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan, 2017.

[120] Jonathon Shlens. A tutorial on principal component analysis, 2014.

[121] Ron Shmelkin, Liar Wolf, and Tomer Friedlander. Generating master faces for dictionary attacks with a network-assisted latent space evolution. In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pages 01–08. IEEE, 2021.

[122] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

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

[124] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[125] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[126] Jerzy Stefanowski. Dealing with data difficulty factors while learning from imbalanced data. In *Challenges in computational statistics and data mining*, pages 333–363. Springer, 2015.

[127] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[128] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2892–2900, 2015.

[129] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[130] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[131] José A. Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness. *Information Sciences*, 247:1–20, 2013.

[132] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[133] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[134] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation, 2017.

[135] Ries Uittenbogaard, Clint Sebastian, Julien Vijverberg, Bas Boom, Dariu M. Gavrila, and Peter H.N. de With. Privacy protection in street-view panoramas using depth and multi-view imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

[136] Fatemeh Vakhshiteh, Ahmad Nickabadi, and Raghavendra Ramachandra. Adversarial attacks against face recognition: A comprehensive study. *IEEE Access*, 9:92735–92756, 2021.

[137] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[138] Pattaramon Vuttipittayamongkol, Eyad Elyan, and Andrei Petrovski. On the class overlap problem in imbalanced data classification. *Knowledge-Based Systems*, 212:106631, 2021.

[139] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018.

[140] Hanchao Wang and Jun Huan. Agan: Towards automated design of generative adversarial networks. *arXiv preprint arXiv:1906.11080*, 2019.

[141] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018.

[142] Xiaosen Wang, Kun He, Chuanbiao Song, Liwei Wang, and John E. Hopcroft. At-gan: An adversarial generator model for non-constrained adversarial examples, 2020.

[143] Mike Wasikowski and Xue-wen Chen. Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1388–1400, 2010.

[144] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks, 2019.

[145] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[146] Xiao Yang, Yinpeng Dong, Tianyu Pang, Jun Zhu, and Hang Su. Towards privacy protection by generating adversarial identity masks, 2020.

[147] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning face representation from scratch, 2014.

[148] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[149] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[150] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[151] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. Alignedreid: Surpassing human-level performance in person re-identification, 2018.

[152] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples, 2018.

[153] Yu-Jun Zheng, Xiao-Han Zhou, Wei-Guo Sheng, Yu Xue, and Sheng-Yong Chen. Generative adversarial network based telecom fraud detection at the receiving bank. *Neural Networks*, 102:78–86, 2018.

[154] Yaoyao Zhong and Weihong Deng. Towards transferable adversarial attack against deep face recognition. *IEEE Transactions on Information Forensics and Security*, 16:1452–1466, 2020.

[155] Sharon Zhou, Mitchell L. Gordon, Ranjay Krishna, Austin Narcomey, Li Fei-Fei, and Michael S. Bernstein. Hype: A benchmark for human eye perceptual evaluation of generative models, 2019.

[156] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

APPENDICES

Appendix A

AutoGAN Appendices

A.1 Implementation Details

The overall structure of the GAN architecture remains consistent across all experiments conducted on tabular data and black and white imagery data. We utilized a fully connected conditional GAN architecture based on the code provided at the following URL: https://github.com/eriklindernoren/Keras-GAN/tree/master/cgan. The primary difference among the GANs lies in the output layer of the generator and the input layer of the discriminator, which are tailored to match the dimensions of the specific dataset being used. The architectures of the generator and discriminator for a dataset with 784 features, specifically the fmnist-based datasets, are presented in Tables A.1 and A.2. For cifar10-based datasets, distinct architectures for the generator and discriminator are employed. The architecture details for the generator and discriminator in the case of cifar10-based datasets are shown in Tables A.3 and A.4, respectively.

Table A.1: The generator architecture utilized for the fmnist-based datasets.

Explanation	Layer	Output
Input1: noise	1	100 Neurons
Input2: class label	1	1 Neuron
transforming class label into 100	2	100 Neurons
multiply transformed class label with input noise	3	100 Neurons
Dense	4	100 Neurons
Dense	5	1024 Neurons
Dense	6	784 Neurons

Table A.2: The discriminator architecture utilized for the fmnist-based datasets.

Explanation	Layer	Output
Input	1	784 Neurons
Dense	2	512 Neurons
Dense	3	512 Neurons
Dense	4	512 Neurons
Dense	5	1 Neuron

Table A.3: The generator architecture utilized for the cifar10-based datasets.

Explanation	Layer	Output
Input1: noise	1	100 Neurons
Input2: class label	1	1 Neuron
Transforming class label into 100	2	100 Neurons
Multiply transformed class label with input noise	3	100 Neurons
Dense	4	4096 Neurons
Reshape to (4, 4, 256)	5	(4, 4, 256)
Conv2DTranspose: filters: 128; kernel shape: (4, 4)	6	(8, 8, 128)
Conv2DTranspose: filters: 128; kernel shape: (4, 4)	7	(16, 16, 128)
Conv2DTranspose: filters: 128; kernel shape: (4, 4)	8	(32, 32, 128)
Conv2D: filters: 3; kernel shape: (3, 3)	9	(32, 32, 3)
Reshape to 3072	10	3072 Neurons

Table A.4: The discriminator architecture utilized for the cifar10-based datasets.

Explanation	Layer	Output
Input1: the image	1	3072 Neurons
Input2: class label	1	1 Neuron
Transforming class label into 100	2	100 Neurons
Multiply transformed class label with input noise	3	100 Neurons
Reshape to (32, 32, 3)	4	(32, 32, 3)
Conv2D: filters: 64; kernel shape: (3, 3)	5	(32, 32, 64)
Conv2D: filters: 128; kernel shape: (3, 3)	6	(16, 16, 128)
Conv2D: filters: 128; kernel shape: (3, 3)	7	(8, 8, 128)
Conv2D: filters: 256; kernel shape: (3, 3)	8	(4, 4, 256)
Flatten	9	4096
Dense	10	1 Neuron

A.1.1 Parameters Used for AutoGAN Algorithm

The hyper-parameters of AutoGAN Algorithm that were used across all the experiments are the following:

- The number of accepted failed attempts: 15;
- Iterations unit: 100;
- The number of generated samples per class for calculating the scores = 500.

The parameters specific to each implemented and tested oracle within the AutoGAN algorithm are as outlined below:

- Oracle CAS_syn:
  - The number of hidden layers for the classifier = 2;
  - The number of perceptrons for the classifier = 100;
  - The number of training epochs for the classifier = 100;
  - Optimizer: adam;
  - Batch size: 32.

- Oracle CAS_real:
  - The number of hidden layers for the classifier = 2;
  - The number of perceptrons for the classifier = 100;
  - The number of training epochs for the classifier = 100;
  - Optimizer: adam;
  - Batch size: 32.
- Oracle CDS:
  - The number of hidden layers for the classifier of CDS = 2;
  - The number of perceptrons for the classifier of CDS = 100;
  - The number of training epochs for the classifier of CDS = 100;
  - Optimizer: adam;
  - Batch size: 32.
- Oracle FCD:
  - The autoencoder consists of 6 layers of sizes: 784, 784×2, 784, 784/2 (the bottleneck), 784×2, 784;
  - Optimizer = 'adam';
  - Loss = 'mse';
  - The number of training epochs for the autoencoder = 200.

A.2 Classifier Details

In our experiments, a fully connected deep neural network serves as the classifier. The number of hidden layers and perceptrons varies for each base dataset, depending on the complexity of the problem. Additionally, the input layer of each neural network is adjusted to match the number of features in the respective datasets. Rectified linear units (ReLU) are employed as the activation functions.

- Tor-based datasets: one hidden layer of 10 perceptrons;
- cic_syscallsbinders_adware-based datasets: two hidden layers of 20 perceptrons;

- cic_syscallsbinders_smsmalware-based datasets: two hidden layers of 20 perceptrons;
- cic_syscalls_adware-based datasets: two hidden layers of 100 perceptrons;
- iscx_spam-based datasets: two hidden layers of 20 perceptrons;
- iscx_defacement: two hidden layers of 100 perceptrons;
- cira-based datasets: one hidden layer of 10 perceptrons;
- mnist-based, fashion-mnist-based , Kuzushiji-mnist-based, and cifar10-based datasets: one hidden layers of five perceptrons.