POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Balancing Image Quality and Attack Effectiveness in Computer Vision

Exploring Generative Adversarial Models for Adversarial Sample Generation

José Areia

School of Management and Technology
Department of Computer Engineering
Master in Cybersecurity and Digital Forensics

Leiria, September 2025

This page intentionally left blank.

POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Balancing Image Quality and Attack Effectiveness in Computer Vision

Exploring Generative Adversarial Models for Adversarial Sample Generation

José Areia
Student No. 2230455

Supervisor: Leonel Santos
Assistant Professor, Department of Computer Engineering, Polytechnic of Leiria

Co-supervisor: Rogério Luís de C. Costa
Assistant Professor, Department of Informatics Engineering, University of Coimbra

School of Management and Technology
Department of Computer Engineering
Master in Cybersecurity and Digital Forensics

Dissertation

Leiria, September 2025

This page intentionally left blank.

Balancing Image Quality and Attack Effectiveness in Computer Vision

Preparation of this work was facilitated by the use of the *IPLeiria-Thesis* template.

This page intentionally left blank.

To God.

This page intentionally left blank.

Acknowledgements

I sincerely thank the Polytechnic of Leiria, particularly the Computer Science and Communication Research Centre (CIIC), for providing the essential resources and a suitable workspace for the development of this thesis. I am also deeply grateful to my supervisors, Professor Rogério Costa and Professor Leonel Santos, for their continuous guidance and support throughout this work. Additionally, I wish to thank the Fundação para a Ciência e a Tecnologia (FCT) for the financial support provided through the grant that enabled me to carry out this work and the project Sustainable Stone by Portugal — Valuing Natural Stone for a Digital, Sustainable and Qualified Future where this work was partially supported by the Sustainable Stone by Portugal agenda funded by European Union/Next Generation EU under Grant 02/C05-i01.02/2022.PC644943391-00000051.

This page intentionally left blank.

Resumo

É bem estabelecido que a inteligência artificial é uma presença duradoura, oferecendo diversas aplicações no nosso quotidiano. No domínio da visão computacional, tais aplicações incluem o reconhecimento facial, a deteção de objetos — utilizados na indústria e em veículos autónomos — bem como a análise de imagens médicas. Contudo, estas aplicações permanecem suscetíveis a vulnerabilidades de segurança, particularmente a ataques adversariais, que introduzem perturbações imperceptíveis capazes de enganar as classificações dos modelos. Com base nestas características, esta dissertação investiga a utilização de modelos generativos para produzir exemplos adversariais capazes de enganar múltiplos modelos de classificação.

Como estudo preliminar, foi utilizado uma *Deep Convolutional Generative Adversarial Network* (DCGAN) com a adição de um codificador para gerar imagens adversariais capazes de enganar cinco modelos de classificação distintos. Posteriormente, foi desenvolvida uma nova arquitetura multiobjetivo, com o nome *Multi-Objective Superstar Adversarial GAN* (MOSA-GAN) concebida para gerar exemplos adversariais ao mesmo tempo que preserva elevada qualidade e fidelidade de imagem. A robustez da MOSA-GAN foi ainda avaliada contra mecanismos de defesa de última geração, para aferir a sua eficácia em contextos adversariais mais amplos. Os experimentos foram conduzidos em conjuntos de dados perturbados por quatro ataques distintos, em cinco níveis de magnitude de perturbação, e avaliados em cinco modelos de classificação. As métricas de desempenho incluíram a *Fooling Rate* (FR), juntamente com métricas que aferem a qualidade de imagem *Fréchet Inception Distance* (FID) e *Learned Perceptual Image Patch Similarity* (LPIPS).

Os resultados indicam que a abordagem inicial atingiu um FR de até 91,21%, mas com fraca qualidade de imagem. Em contraste, a MOSA-GAN alcançou um equilíbrio eficaz, atingindo uma FR de de 89,63% enquanto mantinha elevada qualidade de imagem, com valores de LPIPS e FID tão baixos quanto 0,23 e 0,25, respetivamente. Com defesas, a FR reduziu ligeiramente, enquanto um cenário preservou melhor a qualidade de imagem. Os resultados mostram que modelos generativos são viáveis para gerar imagens adversariais, e que a MOSA-GAN equilibra a eficácia adversarial e a qualidade de imagem, validando a abordagem multiobjetivo, com e sem defesas.

Palavras-Chave: Ataques Adversariais, Ataques Baseados em Perturbações, Defesas Adversariais, Modelos de Aprendizagem Profunda, Redes Generativas Adversariais.

This page intentionally left blank.

Abstract

It is well established that Artificial Intelligence (AI) is an enduring presence, offering diverse applications in daily life. In the domain of Computer Vision (CV), such applications include facial recognition for device unlocking, object detection — employed in industrial settings and autonomous vehicles — and medical image analysis. However, these applications remain susceptible to security vulnerabilities, particularly adversarial attacks, which introduce imperceptible perturbations capable of misleading model classifications. Building on these characteristics, this dissertation investigates the use of generative models to produce adversarial samples that can deceive multiple Deep Neural Network (DNN) models.

As a preliminary study, a Deep Convolutional Generative Adversarial Network (DCGAN) augmented with an encoder was employed to generate adversarial images targeting five distinct DNN models. Subsequently, a novel multi-objective architecture, Multi-Objective Superstar Adversarial GAN (MOSA-GAN), was developed to simultaneously generate adversarial samples while preserving high image quality and fidelity. The robustness of MOSA-GAN was further evaluated against state-of-the-art defence mechanisms to assess its effectiveness in broader adversarial contexts. Experiments were conducted on datasets perturbed by four distinct attacks across five levels of perturbation magnitude and evaluated on five DNN models. Performance metrics included Fooling Rate (FR), alongside image quality measures Fréchet Inception Distance (FID) and Learned Perceptual Image Patch Similarity (LPIPS).

Results indicate that the initial approach achieved a FR of up to 91.21%, but exhibited poor image quality and fidelity. In contrast, MOSA-GAN achieved a balanced trade-off, reaching a FR of 89.63% while maintaining high image quality, with LPIPS and FID scores as low as 0.23 and 0.25, respectively. When defences were applied, FR showed a slight reduction, with a minor deterioration in image quality in one scenario. These findings demonstrate the feasibility of using generative models for adversarial image generation and confirm that MOSA-GAN effectively balances adversarial effectiveness with image fidelity, validating the proposed multi-objective approach in both the presence and absence of defence strategies.

Keywords: Adversarial Attacks, Adversarial Defences, Deep Neural Networks, Generative Adversarial Networks, Perturbation-Based Attacks.

This page intentionally left blank.

Contents

List of Figures x

List of Tables xii

Glossary xiv

Acronyms xvi

Symbols xix

1 Introduction **1**

2 Background and Related Work **4**

2.1 Computer Vision 4

2.2 Adversarial Machine Learning 6

2.3 Generative Adversarial Networks 7

2.3.1 Generative Adversarial Network Variants 8

2.3.2 Encoder Architectures in GANs 10

2.3.3 Multiple-Objective Learning 11

2.4 Adversarial Generation Techniques 12

2.4.1 Fast Gradient Sign Method (FGSM) 13

2.4.2 Universal Adversarial Perturbation (UAP) 13

2.4.3 Truncated Ratio Maximization UAP (TRM-UAP) 14

2.4.4 Stochastic Gradient Aggregation (SGA) 15

2.5 Defence Mechanisms Against Adversarial Attacks 16

2.6 Metrics for Assessing Adversarial Sample Generation 18

2.6.1 Related Work 19

3 DCGAN-Based Framework for Adversarial Sample Generation **21**

3.1 Methodology 21

3.2 Experimental Evaluation Environment 23

3.2.1 Data Collection and Preprocessing 23

3.2.2 Adversarial Attack Synthesis 24

3.2.3 Generative Model Selection and Network Architecture 25

3.2.4 Hyperparameter Selection 27

3.2.5 Evaluation Models, Metrics and Setup 28

3.3 Experimental Evaluation Results and Analysis 29

3.3.1 Adversarial DCGAN Performance 29
3.3.2 Batch Size Impact on Adversarial Samples 32
3.3.3 Evaluating Adversarial Image Quality 32
3.3.4 Discussion of Results 33
3.4 Chapter Summary 34

4 Multi-Objective Generative Model for Adversarial Sample Generation 35
4.1 Design of the Proposed Generative Architecture 35
4.2 Methodology 37
4.3 Experimental Evaluation Environment 39
4.3.1 Network Architecture 39
4.3.2 Hyperparameter Selection 40
4.3.3 Defences Implemented 42
4.3.4 Experimental Setup 42
4.4 Experimental Evaluation Results and Analysis 42
4.4.1 MOSA-GAN Model Performance 43
4.4.2 Influence of Discriminator Weight Variation 44
4.4.3 Robustness of MOSA-GAN Against Defensive Techniques 46
4.4.4 Evaluating Adversarial Image Quality 48
4.4.5 Discussion of Results 50
4.5 Chapter Summary 51

5 Conclusion 52

Bibliography 55

Appendices

A MOSA-GAN Generative Model Code Implementations 68

B Adversarial DCGAN Generative Model Code Implementations 72

This page intentionally left blank.

List of Figures

2.1 Overview of a CNN architecture designed for a classification task. 5
2.2 Overview of a simple GAN architecture. 7
2.3 Application of the FGSM attack. 14

3.1 Top-level architecture of the DCGAN model employed in the initial study. 26
3.2 Performance results for the FR metric. 30
3.3 Performance results for the LPIPS metric. 31
3.4 Performance results for the FID metric. 31
3.5 Results of the batch size analysis. 32
3.6 Comparison of images generated by the DCGAN versus real images. 33

4.1 Proposed MOSA-GAN architecture. 36
4.2 Correlation between average metric results and perturbation magnitudes. 44
4.3 Correlation between image quality metrics and discriminator weighting. 46
4.4 Correlation between average evaluation metrics and the defence type used. 48
4.5 Comparison of images generated using different discriminator weight configurations, including the proposed MOSA-GAN, against real images. 49

This page intentionally left blank.

List of Tables

2.1 Overview of black-box scenario adversarial attacks. 13
2.2 Overview of white-box scenario adversarial attacks. 13

3.1 Class mapping used in the Imagewoof dataset. 24
3.2 Applied image transformations and their corresponding parameters. 24
3.3 Hyperparameters and their values used in the model training process. 27

4.1 Hyperparameters and their values used in the MOSA-GAN training process. 40
4.2 Summary of evaluation metrics for MOSA-GAN performance. 43
4.3 Discriminator weight variations used in the ablation test. 44
4.4 Effect of discriminator weight variation on model performance. 45
4.5 MOSA-GAN robustness of different defence techniques. 47

This page intentionally left blank.

Glossary

Adversary An entity, often modelled as an attacker, that intentionally manipulates input data or exploits model vulnerabilities to mislead, degrade, or subvert the performance of a machine learning system. (*p. 6*)

Fréchet A measure of similarity between curves that takes into account the location and ordering of the points along the curves, commonly known as the Fréchet distance. (*p. 18*)

Nadir Point In multi-objective optimisation, the nadir point is the point in the objective space representing the worst values of each objective among the Pareto-optimal solutions. It is used to understand the extent of trade-offs and to normalise objectives. (*p. 11, 12, 37*)

Nadir Slack In multi-objective optimisation, nadir slack is a small positive value added to the nadir point to slightly relax the worst objective values among Pareto-optimal solutions. It is used to improve numerical stability, define a relaxed objective space, and aid in normalisation or scalarisation of objectives. (*p. 37, 41*)

Pareto Front In multi-objective optimisation, the Pareto Front is the set of non-dominated solutions, where no objective can be improved without degrading at least one other objective. It represents the trade-offs among conflicting objectives. (*p. 11, 12*)

Perturbation A carefully crafted modification to an input, designed to deceive a machine learning model into producing an incorrect output. In the context of adversarial attacks, perturbations are typically constrained to be small under a chosen norm so that they remain imperceptible to humans while maximising the likelihood of model misclassification. (*p. 13, 15, 16*)

Transferability The property of an adversarial perturbation crafted for one model to remain effective in misleading other models, even when these models differ in architecture or training data. (*p. 18*)

This page intentionally left blank.

Acronyms

AI Artificial Intelligence. (*p. v, 1, 2, 52*)
ALI Adversarially Learned Inference. (*p. 10*)

BCE Binary Cross-Entropy. (*p. 28, 41*)
BiGAN Bidirectional GANs. (*p. 10*)

CNN Convolutional Neural Network. (*p. 5, 8, 14, 15, 17, 18*)
CV Computer Vision. (*p. v, 1, 2, 4, 5, 12, 13, 52*)

DA Data Augmentation. (*p. 9, 10, 24*)
DCGAN Deep Convolutional Generative Adversarial Network. (*p. v, 8, 9, 25–27, 29, 32, 34, 45, 46, 72*)
DL Deep Learning. (*p. 4, 5*)
DNN Deep Neural Network. (*p. v, 4, 7, 21, 23, 24, 27, 28, 30, 33, 34, 38, 42, 43, 45, 48, 51–53*)

FGSM Fast Gradient Sign Method. (*p. 12, 13, 24, 25, 27, 29, 30, 40, 43, 44*)
FID Fréchet Inception Distance. (*p. v, 18, 19, 29, 30, 32–34, 43, 44, 46, 48, 50–52*)
FR Fooling Rate. (*p. v, 14, 16–18, 23, 29, 30, 32–34, 38, 42–48, 50–53*)

GAN Generative Adversarial Network. (*p. 7–11, 25*)

HOG Histogram of Oriented Gradient. (*p. 4*)
HV Hypervolume. (*p. 11, 12, 37*)

k-NN k-Nearest Neighbours. (*p. 4*)

LPIPS Learned Perceptual Image Patch Similarity. (*p. v, 18, 19, 29, 30, 32–34, 43, 44, 46, 48, 50–52*)

ML Machine Learning. (*p. 2, 4, 6, 7, 12, 21*)
MOSA-GAN Multi-Objective Superstar Adversarial GAN. (*p. v, 2, 3, 35–44, 46, 47, 49–53, 68*)

NN Neural Network. (*p. 5, 6, 8, 17*)

PGD Projected Gradient Descent. (*p. 17*)

SGA Stochastic Gradient Aggregation. (*p. 12, 16, 24, 25, 29, 30, 32, 43*)
SIFT Scale-Invariant Feature Transform. (*p. 4*)
SVM Support Vector Machine. (*p. 4*)

TRM-UAP Truncated Ratio Maximization UAP. (*p. 12, 14, 15, 24, 25, 29, 30, 32, 33, 43*)

UAP Universal Adversarial Perturbation. (*p. 12–15, 21, 22, 24, 25, 29, 30, 32, 43*)

This page intentionally left blank.

Symbols

C Number of channels of a given image. Commonly represented in the RGB colour model. Each channel value is denoted by $x \in \{0, \ldots, 255\}$. (*p. 4*)

K Number of classes in a given set of images. (*p. 4*)

δ Perturbation applied to an input, typically computed within an adversarial attack algorithm, designed to induce a model misclassification while remaining imperceptible or minimal according to a specified norm. (*p. 13, 14, 16*)

η Represents the Nadir Point associated with the multi-objective optimisation process. (*p. 11*)

H Height of a given image, expressed as a real number in $\mathbb{R}$. (*p. 4*)

σ An activation function, typically applied element-wise in neural networks to introduce non-linearity. Common choices include the sigmoid, hyperbolic tangent, and ReLU. Formally, given an input $x \in \mathbb{R}$, the activation function computes $\sigma(x)$. (*p. 8*)

W Width of a given image, expressed as a real number in $\mathbb{R}$. (*p. 4*)

ξ Magnitude of the perturbation designed for a specific adversarial attack. (*p. 14*)

This page intentionally left blank.

1 Introduction

It is well established that Artificial Intelligence (AI) is here to stay. Numerous benefits can be realised through this technology, including the automation of repetitive tasks (Pannu, 2018), enhanced data analysis and decision-making (Lu, 2019), advancements in healthcare diagnostics and treatment (Racine et al., 2019), and risk prediction and management (Caiming Zhang et al., 2021). One of the main advantages of AI lies in the field of Computer Vision (CV), which involves the creation or analysis of images and videos to improve related tasks, such as generating a desired image or classifying it (Voulodimos et al., 2018). CV applications are widely used in daily life, including facial recognition for device unlocking (Wang et al., 2019), object detection, employed in both industrial settings and autonomous vehicles (Parekh et al., 2022), barcode and QR code scanning (Tiwari, 2016), medical imaging analysis (Racine et al., 2019), and security surveillance (Szeliski, 2022). Recently, the integration of robotics and CV has also begun to emerge. For example, Gemini Robotics employs AI-based CV tools to classify objects, enabling their robots to perform a variety of tasks (Gemini Robotics Team et al., 2025). This type of implementation clearly demonstrates that AI, particularly in CV, is becoming an integral component of modern technology.

However, these tools are often not developed with a security-by-design mindset, and as research continues to advance and CV technologies become increasingly integrated into daily life, the risk of malicious exploitation grows. The potential for misuse is a significant concern, particularly in relation to the everyday lives of users, and the field of CV and its applications is no exception (Chiyu Zhang et al., 2025).

In 2024, a flaw in Apple's Vision Pro headset allowed attackers to interpret users' eye movements and reconstruct typed messages and passwords with high accuracy (Burgess, 2024). This vulnerability exploited the device's eye-tracking feature, raising serious concerns about biometric data privacy. Moreover, in 2025, researchers developed a technique named *RisingAttacK*, which subtly manipulates key features in visual inputs to deceive AI systems without producing visible changes for human observers (Paniagua et al., 2025). This method affects widely used vision architectures such as ResNet-50 and ViTB, posing risks to systems that rely on accurate visual data, including autonomous

vehicles. This type of technique falls within the category of adversarial attacks.

An adversarial attack is a deliberate attempt to manipulate the input of a Machine Learning (ML) model to induce incorrect or unexpected outputs (Huang et al., 2011). In the context of CV, these attacks typically involve small, often imperceptible perturbations to images that cause the model to misclassify them (Finlayson et al., 2019). Despite their minimal nature, such alterations exploit vulnerabilities in the model's feature extraction and decision-making processes, exposing fundamental weaknesses in its robustness. Adversarial attacks are of critical concern in cybersecurity because they can undermine the reliability and safety of AI-driven systems, such as those mentioned previously — autonomous vehicles, biometric authentication, and surveillance systems — potentially resulting in severe operational and security consequences (Chiyu Zhang et al., 2025). Beyond security, adversarial attacks susceptibility raises questions of fairness, transparency, and accountability, as systems may be manipulated in ways that undermine their intended use (Brundage et al., 2018). Therefore, the mitigation of adversarial attacks are essential for ensuring the trustworthiness and resilience of AI technologies.

Building on the need to mitigate adversarial attacks and to ensure the trustworthiness of AI applications in daily life, this dissertation presents the following work. We investigate whether a generative model trained on malicious images can produce on-demand adversarial images with imperceptible modifications that mislead a target classification model. The objectives and contributions of this dissertation are threefold:

- Test the aforementioned hypothesis and develop a toolbox capable of generating adversarial images.
- Analyse the behaviour of the generative model and assess whether it can extract and learn adversarial features from the training data.
- Evaluate the model's potential as a robustness mechanism, *i.e.*, whether the generated n-images can be used to augment classifier training and thereby improve resistance to such attacks.

Given the problem statement and objectives of this dissertation, two distinct studies were conducted. The first employed a pre-existing generative model to evaluate the points aforementioned, representing the preliminary stage of this work. The second study developed a novel generative architecture, MOSA-GAN, informed by insights from the first study, specifically the need to balance attack effectiveness and image quality. Both studies include a detailed methodology and extensive evaluation to ensure rigorous academic and scientific results. Additionally, these studies contribute to the scientific community by presenting a new approach for testing this hypothesis, with comprehensive evaluation of both attack effectiveness and image quality. The first study was presented and published at a conference (Areia et al., 2025c), while the second study was also presented at a conference (Areia et al., 2025a) and subsequently published (Areia et al., 2025b). Importantly, the second study introduces the novel adversarial generative architecture MOSA-GAN, which employs multi-objective optimisation mechanisms designed to address the requirements outlined above. Furthermore, all code and materials

have been made available as open source.

Additionally, all three objectives of this study were successfully achieved, as two distinct investigations were conducted sequentially, each producing strong results aligned with the stated objectives. One of the main secondary objectives, as previously mentioned, was to balance attack effectiveness and image quality using a generative model. The experimental evaluations demonstrated that this is not only possible but feasible, as evidenced by the development of MOSA-GAN.

Beyond this achievement, further work was undertaken to strengthen the contribution under rigorous scientific standards. Experimental tests included variations in perturbation magnitude, application of defences against the proposed model, ablation studies of the multi-objective optimisation framework, and extensive evaluations across both models. Collectively, these efforts confirm that the objectives were not only met but also that the study was thoroughly assessed and its robustness reinforced.

Finally, this dissertation is organised as follows. Chapter 2 provides a theoretical overview of the key topics addressed, followed by Chapter 3, which presents the work developed during the preliminary stage of the research. The subsequent Chapter 4 introduces the main contribution of this study, namely the novel MOSA-GAN model architecture. The dissertation concludes with Chapter 5, where the key findings are summarised and potential directions for future research are discussed.

2

Background and Related Work

This chapter aims to present the fundamental knowledge necessary to understand the concepts discussed in this thesis. It is structured sequentially, beginning with a brief introduction to CV, followed by an overview of adversarial ML. Next, the concepts and variants of generative models are presented, followed by a discussion of adversarial attacks and defences. Finally, this chapter reviews the evaluation metrics used to assess the quality of generated adversarial samples and their impact on a given DNN model, concluding with related work relevant to this dissertation.

2.1 Computer Vision

As ML increasingly permeates various aspects of modern life, CV has become a crucial component in the development of intelligent systems (Chiyu Zhang et al., 2025). The application of Deep Learning (DL) has driven substantial progress in numerous CV tasks, including object detection (Ouyang et al., 2017; Diba et al., 2016), motion tracking (Doulamis, 2018), action recognition (Lin et al., 2016; Cao et al., 2016), human pose estimation (Toshev et al., 2014; X. Chen et al., 2014), and semantic segmentation (Noh et al., 2015; Long et al., 2015). In simple terms, its goal is to extract meaningful information from visual data, allowing machines to perform tasks that typically require human visual understanding (Doulamis, 2018).

Additionally, a common task in CV is image classification, where the objective is to assign a label or category to an input image. Formally, given an image $x \in \mathbb{R}^{H \times W \times C}$, where H, W and C denote height, width, and number of channels, a classification model outputs a predicted label $y \in \{1, 2, \ldots, K\}$ among K possible classes.

Early approaches in CV relied on handcrafted feature extraction techniques, such as Scale-Invariant Feature Transform (SIFT) (Lowe, 2004), Histogram of Oriented Gradient (HOG) (Dalal et al., 2005), and colour histograms, combined with conventional classifiers like Support Vector Machine (SVM) or k-Nearest Neighbours (k-NN). These methods were constrained by their reliance on manually designed features, which often required domain expertise and struggled to generalise in complex or highly variable

visual scenarios (Zhao et al., 2024).

The advent of DL, particularly Convolutional Neural Network (CNN), has fundamentally transformed CV. CNNs were inspired by the structure of the visual system, particularly the models proposed by Hubel et al. (1962). These architectures can automatically learn hierarchical feature representations directly from raw image data, capturing simple edges in shallow layers and complex object structures in deeper layers.

A CNN comprises three primary types of neural layers: *i*) convolutional layers, *ii*) pooling layers, and *iii*) fully connected layers, each serving a distinct function. Figure 2.1 illustrates a CNN architecture for an image classification task. Each layer transforms the input volume into an output volume of neuron activations, ultimately producing a 1D feature vector in the final fully connected layers.

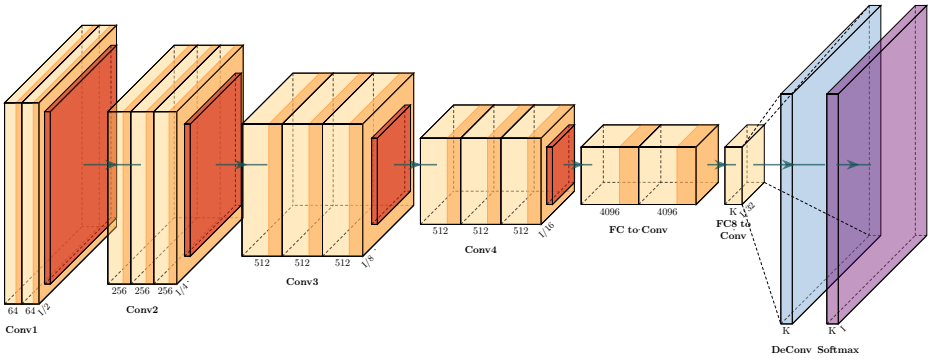


Figure 2.1: *Example architecture of a CNN for a CV task, specifically image classification, comprising convolutional layers, pooling layers, and fully connected layers.*

Well-established architectures, including AlexNet (Krizhevsky et al., 2012), VGG (Simonyan et al., 2015), ResNet (K. He et al., 2016a), and EfficientNet (Tan et al., 2019), have demonstrated state-of-the-art performance in image classification benchmarks.

Regarding the aforementioned widely used application of CV, namely image classification, the typical workflow consists of four steps: *i*) data collection and preprocessing, *ii*) model design, *iii*) model training, and *iv*) model evaluation. In the first step, *data collection and preprocessing*, labelled image datasets are acquired and transformations such as resizing, normalisation, and augmentation are applied to improve model generalisation. In the second step, an appropriate Neural Network (NN) architecture is selected, and the learning objective (*e.g.*, cross-entropy loss) is defined. The third step, *model training*, involves optimising model parameters using gradient-based methods over the training dataset. Finally, in the fourth step, *model evaluation*, performance is assessed on a separate test set using metrics such as accuracy, precision, recall, and F1-score.

Ultimately, understanding the fundamentals of CV, particularly classification tasks, is essential for addressing the cybersecurity challenges in these systems, including adversarial attacks, data poisoning, and model inversion, which exploit the vulnerabilities of visual models (Akhtar et al., 2021).

2.2 Adversarial Machine Learning

According to Huang et al. (2011), adversarial ML involves both the design of ML algorithms capable of resisting sophisticated attacks and the study of the capabilities and limitations of such attacks. These capabilities can be categorised according to the stage at which the attack occurs: the training (poisoning) stage or the testing (evasion) stage (Chiyu Zhang et al., 2025). Each one can further be subdivided into more specific components, as outlined in the list below. The training stage can be divided into three main types, consistent with the definitions provided by Ximeng Liu et al. (2021) and Huang et al. (2011):

- **Data Injection:** The objective is to inject poisoning samples into the training dataset, thereby altering its distribution and causing the learner to train a flawed model. This attack is also referred to as causative poisoning.
- **Data Modification:** This attack aims to contaminate the dataset by modifying it prior to its use in training the target model, *i.e.*, by introducing systematic bias or mislabelling (also known as label flipping).
- **Logic Corruption:** In this attack, the adversary manipulates the learning algorithm, training procedure, or model parameters to insert hidden behaviours (backdoors) or otherwise degrade the model's integrity.

Similarly, the testing stage can also be categorised into three scenarios, as reported by Chiyu Zhang et al. (2025), Bo Liu et al. (2021), and Ximeng Liu et al. (2021):

- **White-box:** The adversary has complete knowledge of the target model, including its architecture, parameters, and training data. Using this information, vulnerabilities are identified and exploited to launch an attack.
- **Black-box:** The adversary has no prior knowledge of the target model. Instead, they infer vulnerabilities by analysing the outputs produced by the model in response to a series of queries.
- **Grey-box:** The adversary has partial knowledge of the target model, such as its architecture or training data, but is unaware of the defence mechanisms in place.

Additionally, adversarial ML comprises several attack types, including model inversion attacks (Al-Rubaie et al., 2019), inference attacks (Bo Liu et al., 2021), poisoning attacks, and model extraction attacks (Ximeng Liu et al., 2021).

In model inversion attacks, certain ML algorithms, such as NNs or ridge regression, do not explicitly store feature vectors. The adversary's objective is to reconstruct feature vectors resembling those used to train the model by exploiting its outputs (Al-Rubaie et al., 2019). Although not always considered a typical adversarial attack, inversion attacks aim to analyse data to illegitimately gain knowledge about a subject (Krumm, 2007; Bo Liu et al., 2021), potentially granting unauthorised access to sensitive information within a model.

Poisoning attacks target the training data to corrupt it, enabling misclassification of specific malicious samples during testing by injecting malicious data, altering labels, or subtly corrupting the dataset (Ximeng Liu et al., 2021), while model extraction attacks aim to recover a target model's parameters, often with black-box access, compromising the confidentiality of the underlying ML algorithm (Chiyu Zhang et al., 2025).

The discovery of adversarial examples reveals that ML models, especially DNNs, are vulnerable to imperceptible perturbations (Szegedy et al., 2014; Ian J Goodfellow et al., 2015), highlighting the instability of their decision boundaries under small input changes (Papernot et al., 2016), and raising concerns about their reliability despite state-of-the-art performance on conventional benchmarks; from a theoretical standpoint, adversarial robustness complements generalisation, where robust models perform well on both natural distributions and adversarially perturbed inputs (Madry et al., 2018), contributing to more secure ML systems and a deeper understanding of learning limitations.

2.3 Generative Adversarial Networks

Generative adversarial models are ML frameworks that create new data samples similar to an existing dataset, targeting the data distribution rather than just classification or regression. Generative Adversarial Networks (GANs) exemplify these models. Introduced by Ian J. Goodfellow et al. (2014), they operate by training two networks in competition. The learning process is framed as a game between two components: a generator ($\mathcal{G}$), which produces synthetic data, and a discriminator ($\mathcal{D}$), which assesses its authenticity (Creswell et al., 2018). This adversarial interaction drives $\mathcal{G}$ to improve iteratively until its outputs are nearly indistinguishable from real data, making generative adversarial models a cornerstone of modern generative modelling techniques. Figure 2.2 illustrates a typical GAN architecture, including the training process and the expected output generated from this training.

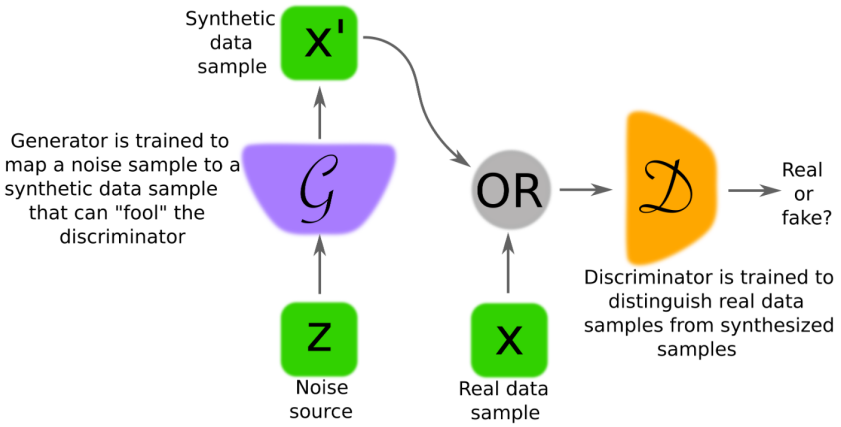


Figure 2.2: *Typical GAN architecture. In this architecture, the two models learned during the training process of a GAN are the discriminator and the generator. Adapted from Creswell et al. (2018).*

As previously mentioned, both networks are trained simultaneously in a competitive

setting. The generator $\mathcal{G}$ learns indirectly, relying on feedback from $\mathcal{D}$, as it does not have direct access to real images. The discriminator $\mathcal{D}$ receives both synthetic samples generated by $\mathcal{G}$ and authentic samples drawn from the real dataset.

In a typical GAN architecture, both $\mathcal{G}$ and $\mathcal{D}$ are generally composed of multi-layer networks, incorporating convolutional and/or fully connected layers (Lee et al., 2019a). The discriminator $\mathcal{D}$ can be defined as a function mapping image data to a probability, indicating whether an image originates from the real data distribution or from $\mathcal{G}$'s distribution: $\mathcal{D} : \mathcal{D}(x) \rightarrow (0, 1)$ (Choi et al., 2018a). For a fixed $\mathcal{G}$, $\mathcal{D}$ is trained to classify images as real (predictions close to 1) or fake (predictions close to 0). Once $\mathcal{D}$ achieves optimal performance, it can be held fixed while $\mathcal{G}$ continues training to reduce $\mathcal{D}$'s classification accuracy (Odena et al., 2017). If $\mathcal{G}$'s distribution eventually matches the real data distribution perfectly, $\mathcal{D}$ becomes maximally uncertain, predicting 0.5 for all inputs. Furthermore, a GAN can be optimised using the following objective functions:

$$\hat{\mathcal{G}} = \arg\min_{\mathcal{G}} \{\mathcal{L}_{adv}(r, \mathcal{D}(\mathcal{G}(Z)))\} \tag{2.1}$$

$$\hat{\mathcal{D}} = \arg\min_{\mathcal{D}} \{\mathcal{L}_{adv}(r, \mathcal{D}(X) + \mathcal{L}_{adv}(f, \mathcal{D}(\mathcal{G}(Z)))\}, \tag{2.2}$$

where r and f are set to one and zero, respectively, Z represents random noise samples from which $\mathcal{G}$ produces synthetic images, and X denotes a batch of real images.

2.3.1 Generative Adversarial Network Variants

While standard GANs can incorporate different NNs architectures, DCGANs specifically employ CNNs in both $\mathcal{G}$ and $\mathcal{D}$ (Alec Radford, 2016). This design exploits the ability of CNNs to capture spatial hierarchies, making DCGANs especially effective for image generation and evaluation (Jenkins et al., 2024; Li et al., 2025).

Within a DCGAN, $\mathcal{G}$ employs transposed convolutional layers — *also referred to as deconvolutional layers* — to upsample an input noise vector Z into a realistic image (Bingqi Liu et al., 2022). The transformation at each layer is defined as:

$$\mathcal{G}(Z) = \sigma(W^T * Z + b), \tag{2.3}$$

where W^T denotes the transposed convolutional filters, $*$ represents the convolution operation, b is the bias term, and σ is an activation function, typically ReLU. Conversely, $\mathcal{D}$ applies convolutional layers to downsample the input image X, extract features, and classify images as real or fake (Alec Radford, 2016). This process is given by:

$$\mathcal{D}(X) = \sigma(W * X + b), \tag{2.4}$$

where W denotes the convolutional filters, with other terms defined as above. The activation function σ differs between the two networks: in $\mathcal{G}$, it is typically ReLU, whereas in $\mathcal{D}$, LeakyReLU is recommended, as suggested by Alec Radford (2016). These convolutional operations allow DCGANs to capture spatial hierarchies in images, resulting in

more realistic (Viola et al., 2021) and coherent (Bingqi Liu et al., 2022) image generation compared to standard GANs that do not exploit such layers.

In addition to DCGANs, several other GAN variants have been proposed, including ControlGAN (Lee et al., 2019b), CycleGAN (Zhu et al., 2020), StarGAN (Choi et al., 2018b), and the more recent SuperstaGAN (Ko et al., 2023).

In ControlGAN, Data Augmentation (DA) is employed together with an additional classifier (C), which is independent of $\mathcal{D}$, to determine whether the generated samples belong to the target domain. A fine-grained classifier trained with DA provides reliable guidance to $\mathcal{D}$, which is optimised using a classification loss derived from this independent classifier. The objective functions of ControlGAN are defined as follows:

$$\hat{\mathcal{G}} = \arg\min_{\mathcal{G}}\{\mathcal{L}_{adv}(r, \mathcal{D}(\mathcal{G}(Z))) + \lambda_{cls} \cdot \mathcal{L}_{cls}(T, C(\mathcal{G}(Z, T)))\} \tag{2.5}$$

$$\hat{\mathcal{D}} = \arg\min_{\mathcal{D}}\{\mathcal{L}_{adv}(r, \mathcal{D}(X) + \mathcal{L}_{adv}(f, \mathcal{D}(\mathcal{G}(Z)))\}, \tag{2.6}$$

$$\hat{C} = \arg\min_{C}\{\mathcal{L}_{cls}(T, C(X_{aug}))\}, \tag{2.7}$$

where λ_{cls} is a hyperparameter for conditional learning, T denotes the target domain label, and X_{aug} denotes the real image modified by DA.

Regarding CycleGAN, the same achieves remarkable results in image-to-image translation in an unsupervised manner, enabling model training without the need for paired datasets by enforcing the property of cycle consistency (Zhou et al., 2016). The CycleGAN framework consists of two generators $\mathcal{G}$ ($\mathcal{G}_{AB}$ and $\mathcal{G}_{BA}$) and two discriminators $\mathcal{D}$ ($\mathcal{D}_A$ and $\mathcal{D}_B$), where each $\mathcal{G}$–$\mathcal{D}$ pair corresponds to a domain (A or B). Given an image X_A from domain A, $\mathcal{G}_{AB}$ translates it into an image in domain B, denoted as $\hat{X}_B = \mathcal{G}_{AB}(X_A)$. Applying both $\mathcal{G}$ sequentially should reconstruct the original image, *i.e.*, $\mathcal{G}_{BA}(\mathcal{G}_{AB}(X_A)) = X_A$. This property, known as cycle consistency, is incorporated into the training process as the cycle-consistency loss. The two discriminators $\mathcal{D}$, $\mathcal{D}_A$ and $\mathcal{D}_B$, then evaluate whether the translated images are realistic and whether they exhibit the features of the corresponding target domain. However, CycleGAN has a fundamental limitation: it can only perform image-to-image translation between two domains (Ko et al., 2023). To address this, StarGAN introduces conditional inputs for $\mathcal{G}$ and an auxiliary classifier C within $\mathcal{D}$. This design enables StarGAN to perform image-to-image translation across multiple domains using a single $\mathcal{G}$–$\mathcal{D}$ pair.

StarGAN's training procedure is similar to the cycle-consistency approach used in CycleGAN. Given an input image X with domain label T, the generator $\mathcal{G}$ translates it to a target domain T', *i.e.*, $\mathcal{G}(X, T')$. The translated image is then reconstructed back into the original domain T using the same $\mathcal{G}$ with the corresponding domain label, *i.e.*, $\mathcal{G}(\mathcal{G}(X, T'), T)$. The reconstructed image is expected to match the original, and an $\mathcal{L}_\infty$ loss is employed to enforce this consistency. This process ensures that the translated image modifies only domain-specific features while preserving the overall content of the original image. The training objectives of StarGAN are defined as:

$$\hat{\mathcal{G}} = \arg\min_{\mathcal{G}}\{\mathcal{L}_{adv}(r, \mathcal{D}(\mathcal{G}(X,T))) + \lambda_{cls} \cdot \mathcal{L}_{cls}(T, C_{\mathcal{D}}(\mathcal{G}(X,T))) + \lambda_{rec} \cdot \mathcal{L}_{rec}(X, \mathcal{G}(\mathcal{G}(X,T),T'))\} \tag{2.8}$$

$$\hat{\mathcal{D}} = \arg\min_{\mathcal{D}}\{\mathcal{L}_{adv}(f, \mathcal{D}(\mathcal{G}(X,T))) + \mathcal{L}_{adv}(r, \mathcal{D}(X)) + \lambda_{cls} \cdot \mathcal{L}_{cls}(T, C_{\mathcal{D}}(X))\}, \tag{2.9}$$

where λ_{cls} and λ_{rec} are hyperparameters controlling the balance of adversarial, classification, and reconstruction losses, and $C_{\mathcal{D}}$ denotes the auxiliary classifier in the $\mathcal{D}$.

In recent years, SuperstarGAN (Ko et al., 2023) was proposed as an extension of StarGAN, incorporating an independent C. This independent C can be trained using DA techniques without interfering with the GAN training process. As a result, C avoids overfitting and is able to capture both domain-invariant and domain-specific features. Consequently, SuperstarGAN can learn mappings across large-scale domains while accurately representing subtle feature variations.

2.3.2 Encoder Architectures in GANs

As discussed in the previous section, GANs consist of a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$. The generator $\mathcal{G}$ learns a mapping from a latent variable $z \sim p_z(z)$, typically sampled from a simple prior distribution such as a Gaussian, to the data space x (Ian J. Goodfellow et al., 2014). Formally, the generator defines the transformation:

$$\mathcal{G} = \mathcal{Z} \to \mathcal{X}, \quad x = \mathcal{G}(z), \tag{2.10}$$

while the discriminator $\mathcal{D}$ is trained to distinguish between true samples $x \sim p_{data}(x)$ and generated ones $\mathcal{G}(z)$. Although this setup enables high-quality sample generation, it does not provide an explicit mechanism to invert the mapping from data space back to the latent space (Berahmand et al., 2024). Such inversion is crucial for applications that require latent representations of data samples, including feature learning, reconstruction, and semantic manipulation (S. Chen et al., 2023). To overcome such limitations, an encoder $\mathcal{E}$ can be introduced, which approximates the inverse mapping

$$\mathcal{E} = \mathcal{X} \to \mathcal{Z}, \quad z = \mathcal{E}(x), \tag{2.11}$$

With the addition of an encoder $\mathcal{E}$, the GAN framework becomes bidirectional. Examples of such architectures include Bidirectional GANs (BiGAN) (Ding et al., 2020) and Adversarially Learned Inference (ALI) (Dumoulin et al., 2017). These models jointly train $\mathcal{G}$, $\mathcal{D}$, and $\mathcal{E}$ by comparing the joint distributions of pairs $(x, \mathcal{E}(x))$ and $(\mathcal{G}(z), z)$. The adversarial objective in BiGAN and ALI can be expressed as:

$$\min_{\mathcal{G},\mathcal{E}} \max_{\mathcal{D}} \mathbb{E}_{x \sim p_{data}}[\log \mathcal{D}(x, \mathcal{E}(x))] + \mathbb{E}_{z \sim p_z}[\log 1 - \mathcal{D}(\mathcal{G}(z), z)] \tag{2.12}$$

In this formulation, the discriminator $\mathcal{D}$ no longer distinguishes solely between real

and fake samples, but instead between pairs of data and latent codes. By enforcing consistency between the joint distributions of data–latent pairs, the encoder $\mathcal{E}$ learns to map data into meaningful latent representations that are aligned with the generator $\mathcal{G}$ (S. Chen et al., 2023).

The use of encoders in training generative models such as GANs offers several advantages. Representation learning is achieved by mapping data into the latent space, enabling the extraction of compact and semantically meaningful features (Alec Radford, 2016). Data reconstruction becomes possible, as real samples can be projected into the latent space and reconstructed through $\mathcal{G}(\mathcal{E}(x))$. In addition, anomaly detection can be performed, since latent representations can reveal deviations from the learned distributions (Diederik P Kingma et al., 2019). Overall, the integration of encoders extends the role of GANs from purely generative models to versatile tools for both generation and representation learning.

2.3.3 Multiple-Objective Learning

Within a GAN architecture, $\mathcal{G}$ and $\mathcal{D}$ have distinct roles and objectives, but their shared ultimate goal is to generate images that closely resemble the original data. In some applications, particularly when employing multiple $\mathcal{D}$s, additional objectives may be required, making multi-objective learning especially relevant (Albuquerque et al., 2019).

In a GAN with multiple $\mathcal{D}$s, $\mathcal{G}$ receives feedback (gradients) from each $\mathcal{D}$, with each $\mathcal{D}$ assessing $\mathcal{G}$'s performance using a distinct loss ℓ_k. Simply averaging these losses, as done in some methods, can dilute feedback from the $\mathcal{D}$s where $\mathcal{G}$ performs poorly. It is therefore preferable to prioritise the *hardest-to-satisfy* $\mathcal{D}$s, where $\mathcal{G}$'s outputs are most easily distinguished from real data (C. He et al., 2020). Hypervolume (HV) maximisation provides a solution to this issue.

In multi-objective optimisation, the Pareto Front set represents solutions that cannot be improved in one objective without degrading another (Van Veldhuizen et al., 1998). HV quantifies the *volume* of the space dominated by these solutions relative to a reference point, called the nadir point and denoted by η (Albuquerque et al., 2019). The optimisation goal is to maximise this HV, thereby improving all objectives (or losses) while maintaining a balance between them (X. Zhang et al., 2023).

As the training process acquires multiple $\mathcal{D}$s, the same provide multiple loss functions ℓ_k for $\mathcal{G}$. Therefore, for each loss function, the mentioned HV maximisation is applied as follows:

- **Objective Vector**: Each loss ℓ_k represents an objective in the multi-objective space.
- **Nadir Point** (η): A reference point serving as an upper bound for the losses. It is dynamically updated during training to reflect the current maximum D loss scaled by a slack parameter $\delta > 1$:

$$\eta = \delta \cdot \max_k(\ell_k) \tag{2.13}$$

- **HV Loss**: Generator $\mathcal{G}$ aims to minimise the negative log-HV. Each term $log(\eta - \ell_k)$ emphasises $\mathcal{D}$s with higher losses, prioritising the *hardest-to-satisfy* objectives. This loss encourages $\mathcal{G}$ to balance improvements across all Ds rather than focusing on a subset:

$$L_{\mathcal{G}} = -\sum_{k=1}^{K} \log(\eta - \ell_k) \tag{2.14}$$

Discriminators $\mathcal{D}$ for which $\mathcal{G}$ performs poorly (*i.e.*, high ℓ_k) exert a stronger influence, as $\frac{1}{\eta - \ell_k}$ becomes larger. By updating the nadir point according to the maximum loss, the model prevents excessive emphasis on smaller losses during training. This strategy leads to a *Pareto-optimal* outcome, ensuring that $\mathcal{G}$ improves consistently across all discriminator objectives and progresses towards the Pareto Front.

2.4 Adversarial Generation Techniques

An adversarial attack is a technique that manipulates ML models by introducing small, often imperceptible perturbations to the input (Szegedy et al., 2014; Ian J Goodfellow et al., 2015), leading the model to produce incorrect predictions (Carlini et al., 2017). Since the introduction of adversarial examples, numerous attack methods have been proposed, broadly categorised into white-box and black-box attacks (Ximeng Liu et al., 2021).

As noted in Section 2.2, adversarial attacks may fall within white-box, grey-box, or black-box settings. Another important distinction, particularly in classification tasks, is whether an attack is targeted or non-targeted (Chakraborty et al., 2021). In a *targeted attack*, the adversary aims to mislead the model into classifying an input as a specific target class of their choice. In contrast, in a *non-targeted attack*, the objective is simply to induce misclassification, without dictating the output class. Since targeted attacks are more difficult to construct and computationally more demanding, non-targeted attacks are more commonly studied in the literature (Ximeng Liu et al., 2021).

Additionally, within the domain of adversarial attacks on CV, several categories of attack methods have been identified, including optimisation-based attacks (Dong et al., 2018), gradient-based attacks (Ian J Goodfellow et al., 2015), transfer-based attacks (Athalye et al., 2018b), decision-based attacks (Brendel et al., 2018), and confidence-based attacks (P.-Y. Chen et al., 2017). Divided into threat models — white-box and black-box scenarios — Table 2.2 and Table 2.1 present an overview of some of the most widely recognised attacks on CV, respectively, in each threat model.

It is important to note that this section does not address all the attacks mentioned previously. Instead, it focuses on four specific attacks: Fast Gradient Sign Method (FGSM), Universal Adversarial Perturbation (UAP), Truncated Ratio Maximization UAP (TRM-UAP), and Stochastic Gradient Aggregation (SGA). The selection is justified as follows: FGSM and UAP are seminal methods, widely cited in the literature and of considerable historical significance, while TRM-UAP and SGA represent more recent advances that deliver state-of-the-art performance.

Table 2.1: *Overview of the most recognised adversarial attacks on CV under the black-box threat model.*

Attack	Target Type	Method
One Pixel Attack (Su et al., 2019)	Non-Targeted	Optimisation-Based
EOT (Athalye et al., 2018b)	Targeted	Transfer-Based
Boundary Attack (Brendel et al., 2018)	Both	Decision-Based
Biased Boundary Attack (Brunner et al., 2019)	Both	Decision-Based
ZOO (P.-Y. Chen et al., 2017)	Both	Confidence-Based
AutoZOOM (Tu et al., 2020)	Both	Confidence-Based

Table 2.2: *Overview of the most recognised adversarial attacks on CV under the white-box threat model.*

Attack	Target Type	Method
L-BGFS (Szegedy et al., 2014)	Targeted	Optimisation-Based
FGSM (Ian J Goodfellow et al., 2015)	Non-Targeted	Gradient-Based
BIM (Kurakin et al., 2017)	Non-Targeted	Gradient-Based
PGD (Madry et al., 2018)	Non-Targeted	Gradient-Based
JSMA (Papernot et al., 2016)	Targeted	Gradient-Based
C&W (Carlini et al., 2017)	Targeted	Gradient-Based
Deepfool (Moosavi-Dezfooli et al., 2016)	Non-Targeted	-
UAP (Moosavi-Dezfooli et al., 2017)	Non-Targeted	Optimisation-Based
SGA-UAP (Xuannan Liu et al., 2023)	Non-Targeted	Optimisation-Based
TRM-UAP (Y. Liu et al., 2023)	Non-Targeted	Optimisation-Based
Randomised Smoothing (Cohen et al., 2019)	Non-Targeted	Optimisation-Based
Obfuscated Gradient Attack (Athalye et al., 2018a)	Targeted	Optimisation-Based

2.4.1 Fast Gradient Sign Method (FGSM)

Szegedy et al. (2014) initially attributed adversarial samples to the non-linearity and overfitting of neural networks. Contradicting this view, Ian J Goodfellow et al. (2015) showed that even simple linear models are susceptible to such samples. Building on this observation, they introduced the FGSM attack. Formally, FGSM is defined as:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f(x), y)), \tag{2.15}$$

where x' is the adversarial example, x is the original input, ϵ is the perturbation magnitude, $\nabla_x \mathcal{L}(f(x), y)$ represents the gradient of the loss function $\mathcal{L}$ with respect to x, and $\text{sign}(\cdot)$ denotes the sign function. Figure 2.3 provides a graphical representation of the FGSM attack.

2.4.2 Universal Adversarial Perturbation (UAP)

Unlike the previous method, which requires generating a unique perturbation for each image in a dataset of size n, Moosavi-Dezfooli et al. (2017) proposed a universal attack called UAP. This approach creates a single perturbation δ that can be applied across multiple images to consistently mislead a given model. The perturbation δ is generated using an algorithm that must satisfy the following constraints:

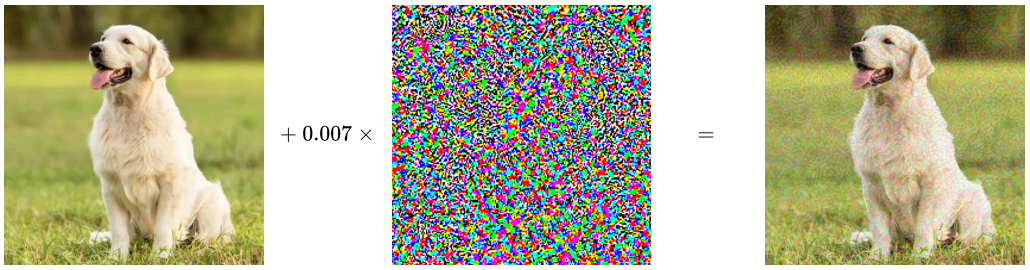


Figure 2.3: *Application of the FGSM attack adapted from the work of Ian J Goodfellow et al. (2015). Starting with the image x, which the classifier initially classifies as "Golden Retriever", a perturbation denoted by the sign$(\cdot)$ function – corresponding to the gradient sign – multiplied by the factor ϵ (with a value of 0.007) is added. This results in a visually identical image x', but the classifier now misclassifies it as "Dingo" with high confidence.*

$$\|v\|_p \leq \xi, \tag{2.16}$$

$$\mathop{\mathbb{P}}_{x\sim\mu}(f(x+v) \neq f(x)) \geq 1-\delta, \tag{2.17}$$

where μ denotes a distribution of images in $\mathbb{R}^d$ and f define the classification function that outputs an estimated label $f(x)$ for each image $x \in \mathbb{R}^d$. The hyperparameter ξ controls the magnitude of the perturbation vector v, while δ, as the universal perturbation, also quantifies the desired FR for all images sampled from the distribution.

The algorithm incrementally constructs a universal perturbation using an iterative approach (Ximeng Liu et al., 2021). In each iteration, it applies the DeepFool attack (Moosavi-Dezfooli et al., 2016) — a prior method developed by the same authors as UAP — to sequentially move all images in the distribution μ toward their respective decision boundaries. After each update, the perturbation is projected onto the ℓ_p ball of radius ϵ to ensure it remains within the prescribed norm constraint.

2.4.3 Truncated Ratio Maximization UAP (TRM-UAP)

Earlier iterations of UAP attacks aim to maximise CNN activations in a purely data-free manner:

$$\max_v \left\|\mathcal{A}^{(i)}(v)\right\|_2, \text{ for } i = 1, 2, \cdots, L \tag{2.18}$$

$$\text{s.t. } \|v\|_\infty \leq \epsilon \tag{2.19}$$

This approach progressively accumulates errors across multiple convolutional layers, increasing the classification loss and ultimately causing the misclassification of perturbed samples $x+\delta$. In contrast, the TRM-UAP method (Y. Liu et al., 2023) not only maximises the activations of convolutional layers but also constructs the universal perturbation δ by maximising the ratio of activations:

$$\max_{\delta} \frac{\left\|C_{+}^{(i)}(\delta)\right\|_2}{\left\|C_{-}^{(i)}(\delta)\right\|_2}, \text{ for } i = 1, 2, \cdots, L \tag{2.20}$$

$$\text{s.t. } \|\delta\|_\infty \leq \epsilon \tag{2.21}$$

Here, the positive and negative activations in the i-th convolutional layer are defined as $C+^{(i)}(\delta) = \max(C^{(i)}(\delta), 0)$ and $C-^{(i)}(\delta) = \min(C^{(i)}(\delta), 0)$, respectively, with their magnitudes measured using the ℓ_2-norm. Furthermore, as previous studies have demonstrated that not all convolutional layers contribute equally to increasing positive activations and the objective loss, Y. Liu et al. (2023) proposed a TRM-UAP method. This method, based on the ratio maximisation described above, aims to enhance both the intensity and transferability of UAPs. It modifies the overall loss function $\mathcal{L}(\delta)$, which is defined by the authors as follows:

$$\begin{aligned} \mathcal{L}(\delta) &= \textstyle\sum_{i=1}^{L} \log \mathcal{L}_{\alpha}^{(i)}(\delta) \\ &= \textstyle\sum_{i=1}^{l'} \log \mathcal{L}_{+}^{(i)}(\delta) - \alpha \cdot \sum_{i=1}^{l''} \log \mathcal{L}_{-}^{(i)}(\delta) + c \\ &= \textstyle\sum_{i=1}^{l'} \log \mathcal{L}_{+}^{(i)}(\delta) - \alpha \cdot \sum_{i=1}^{l''} \log \mathcal{L}_{-}^{(i)}(\delta), \end{aligned} \tag{2.22}$$

where $c = ((1-\alpha) \cdot L - l' + \alpha \cdot l'') \cdot \log \tau$, *i.e.*, a constant. Consequently, as the loss function proposed by the authors has changed, the perturbation δ crafted by the attack must now satisfy the following constraints:

$$\max_{\delta} \sum_{i=1}^{l'} \log \left\|C_{+}^{(i)}(\delta)\right\|_2 - \alpha \cdot \sum_{i=1}^{l''} \log \left\|C_{-}^{(i)}(\delta)\right\|_2, \tag{2.23}$$

$$\text{s.t. } \|\delta\|_\infty \leq \epsilon \tag{2.24}$$

where δ is constrained by the ℓ_∞-norm with bound ϵ. With these modifications, the authors report that the attack produces UAPs with greater intensity, thereby increasing the probability of successful misclassification. Additionally, they claim that the attack exhibits higher transferability across different CNN models compared to earlier data-free UAP methods.

2.4.4 Stochastic Gradient Aggregation (SGA)

In UAP attacks, gradient instability is a common challenge due to the diversity of samples, making it difficult to craft a single perturbation that generalises effectively across different inputs (Ximeng Liu et al., 2021). To mitigate this, the sign operation is commonly used to efficiently generate adversarial examples by relying on the direction of the gradient rather than its magnitude (Ian J Goodfellow et al., 2015). However, Xuannan Liu et al. (2023) observed that the sign operation can introduce substantial optimisation errors when iterative gradients are highly unstable. To illustrate this effect, consider the

gradients at iterations m and $m+1$, denoted as $\tilde{g}m$ and $\tilde{g}m+1$, respectively:

$$\tilde{g}_m = \left[\cdots, \underline{-0.01}, \ 0.10, \ 0.05, \ \underline{0.70}, \cdots\right]^{\mathrm{T}}, \tag{2.25}$$

$$\tilde{g}_{m+1} = \left[\cdots, \underline{1.00}, \ 0.02, \ 0.30, \ \underline{-0.01}, \cdots\right]^{\mathrm{T}}. \tag{2.26}$$

When applying the sign function to accumulate perturbations δ, large positive values in the right region of $\tilde{g}_m$ can be cancelled out by small negative values in $\tilde{g}_{m+1}$:

$$\operatorname{sign}(\tilde{g}_m) = \left[\cdots, \underline{-1}, \ 1, \ 1, \ \underline{1}, \cdots\right]^{\mathrm{T}}, \tag{2.27}$$

$$\operatorname{sign}(\tilde{g}_{m+1}) = \left[\cdots, \underline{1}, \ 1, \ 1, \ \underline{-1}, \cdots\right]^{\mathrm{T}}, \tag{2.28}$$

$$\begin{aligned} \delta &= \alpha \cdot \operatorname{sign}(\tilde{g}_m) + \alpha \cdot \operatorname{sign}(\tilde{g}_{m+1}) \\ &= \alpha \cdot \left[\cdots, \ \underline{0}, \ 2, \ 2, \ \underline{0}, \cdots\right]^{\mathrm{T}} \end{aligned} \tag{2.29}$$

To address this issue, the authors term it *gradient vanishing*, which occurs due to the combined effects of gradient instability and sign operations. To mitigate this problem, Xuannan Liu et al. (2023) propose SGA, which aggregates multi-step noisy forward gradients before applying a single-step quantisation update at each iteration. First, multiple small-batch samples are randomly selected from a large batch to perform a pre-search, updating the inner adversarial perturbation. The stochastic gradients $\tilde{g}_m$ are then accumulated to update the aggregated gradient g^{Aggs}:

$$g^{\mathrm{Aggs}} \leftarrow g^{\mathrm{Aggs}} + \tilde{g}_m \tag{2.30}$$

After aggregating the inner gradients into a single-step gradient for the outer iteration, the outer adversarial perturbation δ is updated as follows:

$$\delta \leftarrow \mathrm{Clip}_{\delta}^{\epsilon}(\delta + \alpha \cdot \operatorname{sign}(g^{\mathrm{Aggs}})), \tag{2.31}$$

where α denotes the step size, and the $\mathrm{Clip}(\cdot)$ operation constrains the perturbation within the ℓ_∞ norm. Experimental results show that this method significantly improves the generalisation of baseline approaches across different settings, yielding a higher average FR than previously developed attacks.

2.5 Defence Mechanisms Against Adversarial Attacks

In the previous section, the robustness, vulnerabilities, and recent advancements in adversarial attacks were outlined. This section aims to present some of the well-known defence methods in the literature that are capable of mitigating such vulnerabilities.

Adversarial Training

Adversarial training is one of the most widely studied defence mechanisms. The key idea is to incorporate adversarial examples into the training process to improve model robustness. Formally, the objective function can be expressed as a min-max optimisation problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} \mathcal{L}(f_\theta(x+\delta), y) \right], \tag{2.32}$$

where θ denotes the model parameters, $\mathcal{D}$ is the data distribution, δ represents the perturbation constrained to a set $\mathcal{S}$, and $\mathcal{L}$ is the loss function (Madry et al., 2018). This straightforward adversarial defence effectively reduces the FR. State-of-the-art attacks such as Projected Gradient Descent (PGD) (Madry et al., 2018) even recommend adversarial training as a countermeasure to the very attack they propose. TRADES is another notable example, balancing natural accuracy and robustness through a regularised loss function (Zhu et al., 2020).

Modify the Training Process

Beyond adversarial training, several methods adapt the training process to enhance robustness. Defensive distillation leverages softened labels from a teacher network to reduce model sensitivity to perturbations (Papernot et al., 2016). Regularisation-based techniques, such as Jacobian regularisation, penalise excessive sensitivity of model outputs to input variations (Jakubovitz et al., 2018). Label smoothing, which modifies target probabilities during training, has also demonstrated improvements in robustness (Szegedy et al., 2016). More recently, curriculum adversarial training progressively increases attack strength throughout training to improve stability (Cai et al., 2018).

Use of Supplementary Networks

Supplementary networks can serve as pre-processors or detectors to counter adversarial perturbations. MagNet, for instance, incorporates detector and reformer networks to identify adversarial inputs and reconstruct cleaner representations before classification (Meng et al., 2017). Feature squeezing reduces input complexity, such as by lowering colour depth, to limit adversarial noise, often in combination with detector models (W. Xu et al., 2018). Generative approaches, including GAN-based defences, aim to project adversarial examples back onto the data manifold (Samangouei et al., 2018). Similarly, auxiliary classifiers trained alongside the main model have been employed to detect inconsistencies introduced by adversarial perturbations (Gong et al., 2017).

Change Network Architecture

Altering the network architecture can also improve adversarial robustness. Networks incorporating radial basis functions in hidden layers demonstrate greater resistance to perturbations due to their localised responses (H. Xu et al., 2020). Capsule networks, leveraging dynamic routing, have shown enhanced robustness compared to conventional CNNs (Hinton et al., 2018). Architectural elements such as residual connections and batch normalisation have further been linked to improved adversarial stability (K. He et al., 2016b). More recently, Bayesian NNs have been investigated for their capacity to model uncertainty, thereby strengthening defence against adversarial inputs (Lakshminarayanan et al., 2017).

Adversarial Purification

Adversarial purification focuses on removing perturbations from inputs prior to classification. PixelDefend projects inputs onto the training data distribution using generative models (Song et al., 2018). Diffusion-based purification employs iterative denoising to eliminate adversarial noise (Nie et al., 2022). Randomised smoothing offers certified robustness by averaging predictions over Gaussian-perturbed inputs, providing guarantees under ℓ_2-bounded attacks (Cohen et al., 2019). Denoising auto-encoders have also been extensively applied to reconstruct clean samples from adversarially corrupted inputs (Gu et al., 2015).

2.6 Metrics for Assessing Adversarial Sample Generation

To validate successful generation of adversarial samples, some evaluation metrics should be considered, notably FR, FID (Heusel et al., 2017) and LPIPS (R. Zhang et al., 2018). Additionally, transferability — the ability of adversarial perturbations to transfer across models and datasets — should be assessed.

Fooling Rate (FR)

Regarding FR, expressed as a percentage, it denotes the proportion of misclassifications against a given model. Higher values indicate a more effective attack, as it fools a larger number of images. However, false positives must be taken into consideration: first verify that the original images x are correctly classified by the model; only then evaluate their corresponding adversarial images x'. This procedure reduces spurious results and yields a more reliable measure of attack success. Adversarial image generation creates samples with features from a reference set but visually different from the originals, so image quality can not be judged by simple pixel-wise comparison with training images. Metrics such as FID (Heusel et al., 2017) and LPIPS (R. Zhang et al., 2018) should be employed.

Fréchet Inception Distance (FID)

The FID metric compares the mean and covariance of feature representations extracted from the deepest layer of the Inception v3 network (Szegedy et al., 2015). These layers are designed to capture high-level semantic features closely aligned with real-world object representations in a CNN model (Heusel et al., 2017). The metric assesses similarity between two image sets by quantifying how frequently the same high-level features appear in both. After processing all images through the Inception network, the means and covariances of the final-layer activations are compared using the Fréchet distance between two multivariate Gaussian distributions:

$$\mathrm{FID} = \left\|\mu_r - \mu_g\right\| + \mathrm{T_r}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}), \tag{2.33}$$

where μ_r and μ_g denote the means of the real and generated image distributions, respectively, and Σ_r and Σ_g represent their covariance matrices. A lower FID score indicates that the generated images are more similar to the real images in terms of feature distributions.

Learned Perceptual Image Patch Similarity (LPIPS)

Unlike FID, the LPIPS metric assesses perceptual similarity between two individual images rather than between image distributions or sets (R. Zhang et al., 2018). Similar to FID, LPIPS utilises features extracted from a pre-trained deep neural network, computing a weighted distance between the deep feature representations of the two images.

Let F_i and F_j be the feature maps from layer l for images I_i and I_j, respectively. These feature maps are normalised along the channel dimension to ensure each feature map has unit norm:

$$\hat{F}_{ij} = \frac{F_{ij}}{\| F_{ij} \|_2} \tag{2.34}$$

Once normalised, the distance between the two images is computed as:

$$d^l = \frac{1}{H_l W_l} \sum_{h,w} \left\| \hat{F}^l_{i,hw} - \hat{F}^l_{j,hw} \right\|_2^2, \tag{2.35}$$

where H_l and W_l denote the spatial dimensions of the feature maps at layer l. The final LPIPS score is obtained as a weighted sum of the distances across all layers:

$$\text{LPIPS}(I_i, J_i) = \sum_l w_l \cdot d^l \tag{2.36}$$

The weights w_l are typically learned from human perceptual judgements, making LPIPS better aligned with human visual perception compared to traditional pixel-based metrics.

2.6.1 Related Work

Generative models have been shown to embed adversarial traits capable of misleading classification systems, either at the level of individual instances or across entire datasets. Early work by Mopuri et al. (2018) introduced the Network for Adversary Generation (NAG), which leverages generative methods to produce diverse and transferable perturbations. Unlike approaches that focus on crafting a single perturbation, NAG generates a variety of adversarial examples, demonstrating improved generalisability across multiple models.

Expanding on this idea, Poursaeed et al. (2018) proposed Generative Adversarial Perturbations (GAP), a framework capable of producing both universal and image-specific perturbations. GAP was evaluated in classification and segmentation tasks, achieving a fooling rate of over 80% on the widely used pre-trained VGG-16 model. This work highlighted the scalability of adversarial perturbations beyond simple classification problems, showing their effectiveness in more complex visual tasks.

Following this, Xiao et al. (2019) developed AdvGAN, a generative model that incorporates adversarial loss to directly learn perturbations tailored to mislead classifiers. AdvGAN distinguished itself by achieving real-time performance in generating adversarial examples, making it a practical tool for large-scale evaluations of model robustness.

More recently, Sun et al. (2024) investigated the robustness of GAN-based image fusion methods. Their study revealed that even subtle adversarial perturbations could drastically reduce the quality of fused images and impair the performance of downstream tasks. These findings underscored the fragility of generative models when applied to safety-critical domains.

Parallel to these developments, research has also explored the use of multi-objective learning in generative models. Durugkar et al. (2017) introduced Generative Multi-Adversarial Networks (GMAN), where the generator is trained against a Softmax-weighted arithmetic average of K discriminators. Their experiments showed that even a simple averaging strategy among discriminators could yield strong results across diverse evaluation metrics, suggesting that multiple discriminators help stabilise training and improve output quality.

Later, Albuquerque et al. (2019) advanced this idea by applying a multiple gradient descent method with hypervolume maximisation. This approach enabled a more effective trade-off between image quality and computational cost, addressing one of the main limitations of earlier adversarial training frameworks. Their results demonstrated that multi-objective optimisation can guide generative models towards more balanced and efficient solutions, making them more adaptable to real-world constraints.

3

DCGAN-Based Framework for Adversarial Sample Generation

The purpose of this chapter is to present the work developed during the early stages of this study. Given that adversarial perturbations have emerged as a critical issue in the security and privacy of ML models, and that generative artificial intelligence also plays a significant role in this domain, we aimed to design a study where an existing generative model architecture could learn the characteristics of various adversarial attacks and generate perturbations capable of deceiving multiple DNNs. Building on this idea, this chapter discusses the proposed methodology, the experimental environment developed for this purpose, and the results obtained. Additionally, one section is dedicated to presenting and discussing the results of the experimental tests, followed by a summary section that outlines the overall findings and contributions of the study. The contents of this chapter were presented at the *20th International Conference on Availability, Reliability and Security — ARES 2025* (Areia et al., 2025c). Additionally, this work is publicly available in the official GitHub repository[1].

3.1 Methodology

As stated earlier, generative models produce images based on features learned through a competitively trained process (see Section 2.3). Separately, we have noted that adversarial attacks apply perturbations directly to images to compromise a given DNN model (see Section 2.2). Combining these two perspectives, we hypothesise that a generative model, independent of any specific architecture, can learn the characteristics of an adversarial dataset and subsequently generate adversarial images on demand. However, it is important to note that, unlike prior UAP-based or instance-specific approaches, our method does not focus solely on generating a perturbation δ; rather, it trains a generative model to produce transferable adversarial images on demand for multiple target models. For this purpose, we followed a simple five-stage workflow, described below:

[1] GitHub repository: `https://github.com/ipleiria-ciic/adversarial-dcgan`

1. **Data Acquisition and Preprocessing:** Collection of relevant images from the given dataset, followed by cleaning, formatting, and normalisation to ensure consistency and suitability for subsequent stages.
2. **Adversarial Attack Synthesis:** Generation of controlled adversarial samples and perturbations, intended to serve as the primary dataset for subsequent stages.
3. **Generative Model Selection and Training:** Selection and training of a generative model to capture data distributions and produce synthetic adversarial samples.
4. **Representation Encoder Training:** Training of an encoder to learn meaningful feature representations from both original and adversarially generated data.
5. **Performance Testing and Evaluation:** Comprehensive assessment of the trained models through testing, using quantitative metrics and qualitative analysis to evaluate effectiveness and robustness.

Data Acquisition and Preprocessing

In this stage, an existing dataset from the literature was utilised. This dataset serves as a reference for future evaluations and as the basis for generating adversarial images. Additionally, it is important that the dataset is representative and multiclass-labelled to enable robust evaluation. Preprocessing techniques were applied to ensure data uniformity for the subsequent stages.

Adversarial Attack Synthesis

The generation of adversarial attacks requires a clear rationale; it is essential to define the objectives beforehand. Therefore, adversarial attacks must be carefully selected and curated. With this in mind, we chose four different attacks: one instance-specific and three UAP-based attacks. This selection allows us to compare the effectiveness of perturbations from an instance-specific attack against those from universal attacks. Furthermore, to assess the effect of different perturbation magnitudes ξ, separate adversarial datasets were generated for each attack, with each dataset corresponding to a specific perturbation magnitude. The resulting adversarial images were organised by attack type, forming a diverse n-set of perturbed images.

Generative Model Selection and Training

At this stage, we selected the architecture for our generative model. It is important to note that we did not develop a novel architecture; instead, we aimed to test the hypothesis using a well-established, prime generative model. In addition to these requirements, the model needed to produce high-quality images, ensure stable training, and support effective hierarchical feature learning, which guided our architectural choices for both the generator ($\mathcal{G}$) and discriminator ($\mathcal{D}$).

Representation Encoder Training

Because the generative model produces images from a latent noise vector (z), and the primary objective of this study is not to generate random images resembling the originals — which could lead to repeated outputs and evaluation errors — but to accurately mimic real images, an encoder ($\mathcal{E}$) is essential. Specifically, $\mathcal{E}$ learns a structured latent

representation of the real data from the dataset, enabling $\mathcal{G}$ to produce more realistic and diverse outputs. Consequently, instead of generating an adversarial image y as $y = \mathcal{G}(z)$, we generate it as $y = \mathcal{G}(\mathcal{E}(x))$, where $\mathcal{E}(x)$ is the output of $\mathcal{E}$ for a given real image x.

Performance Testing and Evaluation

For this stage, and based on previous studies (Ximeng Liu et al., 2021; Bingqi Liu et al., 2022; Xuannan Liu et al., 2023; Ko et al., 2023), we aimed to assess the FR of the adversarial images against DNN models. To this end, we trained five well-known DNN models for a classification task using the dataset selected for this study. The adversarial images were then evaluated against these models, producing FR values for each attack–perturbation pair. Additionally, our focus was not solely on FR; we also sought high image quality and fidelity to the original images. To this end, appropriate evaluation metrics were applied alongside qualitative assessment by human observers.

3.2 Experimental Evaluation Environment

This section presents the experimental environment used in this study, including the rationale behind each choice, ensuring alignment with the each stage of the methodology described in the preceding section.

3.2.1 Data Collection and Preprocessing

Aligned with the considerations regarding the dataset and previous studies (Mopuri et al., 2017; H. Xu et al., 2020; Y. Liu et al., 2023; Xuannan Liu et al., 2023), we selected a subset of the widely known ImageNet dataset (Russakovsky et al., 2015), specifically the Imagewoof dataset (FastAI, 2020). This dataset contains images from 10 different dog breeds (classes) and is available at resolutions of 320px or 160px; for this study, we used the 160px version. Approximately 10 000 images were used for training, with around 5 000 reserved for validation. We chose this subset rather than the full ImageNet dataset, which has 100 classes, primarily because the larger number of classes decreases the difficulty of fooling the model. This selection provided an optimal trade-off between image quality, usability, and scientific rigour for our study.

Because the chosen subset follows the naming conventions of the ImageNet dataset — *i.e.*, the letter "n" followed by eight digits — and is difficult to interpret manually, especially without proper notation (*e.g.*, a JSON file), these classes were mapped to their corresponding dog breed names as listed in the official ImageNet repository. Table 3.1 presents the mapping used in this study.

As mentioned above, and to ensure consistency and suitability for the subsequent stages, as well as alignment with previous studies, several preprocessing techniques were applied. These included resizing the images to a specific dimension and normalisation using the mean and standard deviation across the three RGB channels. The normalisation followed the approach of the dataset authors (Russakovsky et al., 2015), who explain that normalising input data improves the training process by making it faster

Table 3.1: *Class mapping used in the Imagewoof dataset.*

Numeric Label	Dog Breed Class
n02086240	Shih-Tzu
n02087394	Rhodesian Ridgeback
n02088364	Beagle
n02089973	English Foxhound
n02093754	Border Terrier
n02096294	Australian Terrier
n02099601	Golden Retriever
n02105641	Old English Sheepdog
n02111889	Samoyed
n02115641	Dingo

and more stable, while also preventing vanishing or exploding gradients. Resizing was necessary due to the requirements of the generative model architecture detailed in Section 3.2.3.

In addition to these preprocessing techniques, DA techniques were applied, namely random horizontal flips and random conversion to greyscale. These augmentations were implemented to increase the robustness of the DNN models trained for the classification tasks during the testing and evaluation stage. All transformations were implemented using PyTorch utilities. In Table 3.2, presented below, it is possible to observe the parameters associated with each applied transformation.

Table 3.2: *Applied image transformations and their corresponding parameters.*

Transformations	Values Applied
Resizing	$W = 64$ x $H = 64$
Random horizontal flips	$1\ (True)$
Random greyscale	$P = 0.1$, where $P \Rightarrow Probability$
Mean normalisation	$R = 0.485$, $G = 0.456$, $B = 0.406$
Standard deviation normalisation	$R = 0.229$, $G = 0.224$, $B = 0.225$

3.2.2 Adversarial Attack Synthesis

As outlined in Section 3.1, the images required to train the generative model are not the clean images from the dataset but images containing a perturbation δ produced by an adversarial attack; these images are denoted as y. To generate y, four adversarial attacks, as described in the methodology above, were selected: one instance-specific attack and three UAP-based attacks. For the instance-specific attack we used the simple yet powerful FGSM (Ian J Goodfellow et al., 2015). For the UAP-based attacks we selected the original UAP (Moosavi-Dezfooli et al., 2017) and two recent, strong variants, TRM-UAP (Y. Liu et al., 2023) and SGA (Xuannan Liu et al., 2023).

To assess the impact of perturbation magnitude, five values of ξ were used: $\xi \in \{0.01, 0.05, 0.10, 0.15, 0.20\}$. The choice of this range follows prior work that commonly uses $\xi = 0.10$ as a reference; this value was taken as the central point and two smaller

and two larger magnitudes were chosen with increments of 0.05.

It is also important to note that the application of δ to a given image x depends entirely on the specific attack and its methodology. For both FGSM and UAP, the precomputed δ is applied to each image in an iterative loop; during this process δ is scaled by the perturbation magnitude ξ to achieve the intended adversarial strength. For the TRM-UAP attack, the perturbation δ is first trimmed using a clamping operation:

$$\delta' = min(max(\delta, \xi min), \xi max), \tag{3.1}$$

$$\text{where } \xi min = -10/255, \xi max = 10/255 \tag{3.2}$$

Next, it is added to each individual batch of images:

$$y = \delta' + \alpha \cdot x, \text{where } \alpha = 1 \tag{3.3}$$

And finally, a new clamping operation is applied:

$$y' = min(max(y, \xi min), \xi max), \tag{3.4}$$

$$\text{where } \xi min = 0, \xi max = 1 \tag{3.5}$$

Similarly, for the SGA attack, the perturbation δ is applied and the resulting images are clamped to the $[0, 1]$ range, yielding the adversarial images y.

3.2.3 Generative Model Selection and Network Architecture

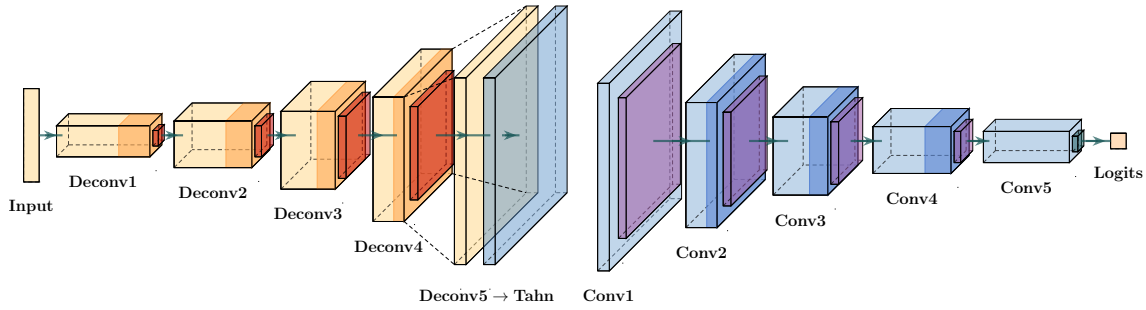
During the development of this study, and as outlined in the methodology (see Section 3.1), we aimed to utilise an existing generative model rather than designing a novel one, in order to focus on testing our hypothesis. Below, we outline the process of selecting this model and the corresponding network architecture it inherits.

Generative Model Selection

Initially, we opted for a simple GAN. However, after preliminary experiments with the chosen dataset and attacks, it became clear that the model would not fulfil our primary objective due to poor image quality and highly unstable training. Consequently, we adopted a more recent variant of the vanilla GAN, namely the DCGAN. From a theoretical perspective, this choice is well justified. Compared to a standard GAN, a DCGAN offers more stable training by employing convolutional layers instead of fully connected layers, thereby improving gradient flow and reducing mode collapse. Moreover, DCGAN is capable of generating higher-quality images, as convolutional layers capture spatial hierarchies more effectively. Finally, the architecture is better suited for handling larger image sizes, since convolutional layers efficiently process local spatial patterns — a significant advantage for subsequent human observation analysis during the testing and evaluation stage.

Network Architecture

With the generative architecture selected (DCGAN), the network follows the corresponding design and comprises a discriminator $\mathcal{D}$ and a generator $\mathcal{G}$. $\mathcal{D}$ employs a deep convolutional architecture of five convolutional layers with progressively increasing channel depth. Each layer is followed by a non-linear activation — LeakyReLU for all layers except the final one, which uses Sigmoid — and batch normalisation is applied to stabilise training, except in the first and last layers. $\mathcal{D}$ accepts a $3\times64\times64$ image as input and processes it through the convolutional stack, producing an output in $[0, 1]$ via a final Sigmoid activation. Padding of 1 is used throughout $\mathcal{D}$ except at the final layer; all convolutional layers use stride 2 and a kernel size of 4. $\mathcal{G}$ consists of five convolutional layers, matching the depth of $\mathcal{D}$. It takes as input a latent vector z of size 100. The first layer employs 512 filters, which are progressively reduced to 64 across the subsequent four layers. The final layer contains 3 filters, corresponding to the image channels, and applies a Tanh activation to produce the generated image. The stride, padding, and kernel size are identical to those used in $\mathcal{D}$. A visual representation of this architecture, without any parameters, is shown below in Figure 3.1.



(**a**) *Generator architecture.* (**b**) *Discriminator architecture.*

Figure 3.1: *In the generator architecture, bright orange represents the convolutional layers, followed by darker orange batch normalisation layers, and red indicates the ReLU activation function applied to all but the last layer. The final blue layer uses the Tanh activation function to scale the output within the* $[-1, 1]$ *range. In the discriminator network, bright blue represents the convolutional layers, followed by darker blue batch normalisation layers, and purple indicates the LeakyReLU activation function applied to all but the last layer. The final bright green layer uses the Sigmoid activation function to output a probability in the range* $[0, 1]$*, indicating whether the image is real or fake.*

Additionally, beyond the original network design, an encoder $\mathcal{E}$ was incorporated to meet the requirements outlined in the methodology. Architecturally, $\mathcal{E}$ is similar to $\mathcal{D}$, as its purpose is to learn the latent representation of a real image. Specifically, it takes a $3 \times 64 \times 64$ image as input and processes it through five convolutional layers, identical to those in $\mathcal{D}$ with the same parameter settings. The feature maps, padding values, kernel size, and stride are consistent with those used in the other two components of the architecture.

It is important to note that no modifications were made to the architecture inherited from the selected generative model, in order to preserve its original properties as outlined at the beginning of this methodology.

3.2.4 Hyperparameter Selection

The selection of hyperparameters represented one of the most time-consuming aspects of this study. Determining optimal values for configurations in both a DNN and a DCGAN is highly challenging and requires considerable effort. Nevertheless, this process was essential to maintain scientific rigour and methodological consistency. For a clear overview of the hyperparameters controlled during the model training, Table 3.3 is presented, detailing each hyperparameter and the final values chosen.

Table 3.3: *The final hyperparameters and their corresponding values used in the model training process.*

Hyperparameter	**Final Setting**
Batch size	128
Input image resolution	64×64
Input channels	3
Epochs	50
Latent vector dimension (z)	100
Feature maps in generator ($\mathcal{G}$)	64
Feature maps in discriminator ($\mathcal{D}$)	64
Feature maps in encoder ($\mathcal{E}$)	64
Adam learning rate (ℓ_r)	2e−4
Adam β_1	0.5
Adam β_2	0.999
ReduceLROnPlateau reduction factor (rd_f)	0.8
ReduceLROnPlateau patience (p)	30
ReduceLROnPlateau minimum learning rate ($min\ \ell_r$)	1e−10

All of these hyperparameters were carefully curated and not arbitrarily chosen at the outset. It is important to note that the testing scenario was based on the FGSM attack dataset with a perturbation magnitude of $\xi = 0.10$.

A lower batch size was initially selected due to the reduced memory usage of the GPU (see Section 3.2.5). However, smaller batch sizes produced weak results, with highly noisy gradients and slow training. Consequently, the batch size was gradually increased up to 128, which yielded the best results in terms of training speed and gradient stability. Regarding the image resolution and input channels, these were necessarily set to 64×64 and 3, respectively, to meet the requirements of the generative architecture model; therefore, no changes were made to this parameter.

The number of epochs was also carefully evaluated through multiple experiments to determine the most practical value. Tests were conducted with 25, 50, 100, 150, and 200 epochs. During these experiments, the losses of $\mathcal{G}$ and $\mathcal{D}$ were closely monitored, and it was observed that in all cases training became unstable around epochs 40–45, with the generator loss diverging. One might argue that other hyperparameters required adjustment to address this issue; therefore, parameters such as feature maps, learning rates (ℓ_r), and beta values of the optimisation function were modified. However, no improvement in stability was achieved beyond this point. Consequently, the final number of epochs was set to 50, as no further training benefits could be obtained after this stage.

Regarding the latent vector dimension z and the feature maps for $\mathcal{G}$, $\mathcal{D}$, and $\mathcal{E}$, the values chosen were aligned with the conventions of the architecture model and prior work. The latent vector was set to $z = 100$, following the empirical conventions established by Ian J. Goodfellow et al. (2014) and Alec Radford (2016). In addition, the feature maps for all components were set to 64, ensuring consistency with the image resolution defined for the training process.

For optimisation, the Adam optimiser was employed (Diederik P. Kingma et al., 2017). This required defining a learning rate ℓ_r and two momentum parameters, β_1 and β_2. While the literature provides baseline values for these parameters, they can be fine-tuned for specific cases and training conditions. Manually adjusting these values, however, is time-consuming and may lead to incomplete assessment. To address this, we employed Optuna (Akiba et al., 2019), a tool that automates the search for optimal hyperparameters. Optuna evaluates a predefined range of values for each parameter, analyses the losses of the model components, and determines the configuration that minimises the loss.

For the Adam optimiser, we defined the following search spaces for the test scenario: $\ell_r = \{1e{-}4, 2e{-}4, 3e{-}4, 4e{-}4, 5e{-}4\}$, $\beta_1 = \{0.40, 0.45, 0.50, 0.55, 0.60\}$, and $\beta_2 = \{0.97, 0.98, 0.99, 0.995, 0.999\}$. After running the Optuna optimisation, the best values obtained were $\ell_r = 2e{-}4$, $\beta_1 = 0.50$, and $\beta_2 = 0.999$.

For additional optimisation and learning rate control, the ReduceLROnPlateau scheduler (Paszke et al., 2019) was used. This scheduler automatically reduces the learning rate when a monitored metric stops improving, thereby enhancing training efficiency. The reduction factor, patience, and minimum learning rate ($min\ \ell_r$) can be adjusted. Similar to the Adam optimiser, these parameters were optimised using the Optuna tool. The search spaces were defined as follows: $rd_f = \{0.5, 0.6, 0.7, 0.8, 0.9\}$, $p = \{10, 20, 30, 40, 50\}$, and $min\ \ell_r = \{1e{-}4, 1e{-}5, 1e{-}6\}$. After optimisation, the best values obtained were $rd_f = 0.8$, $p = 30$, and $min\ \ell_r = 1e{-}6$.

Additionally, although no configuration was required for this function, it is important to note that Binary Cross-Entropy (BCE) was used, as it is the canonical loss function employed in the original work of Ian J. Goodfellow et al. (2014). This loss function is essential because it penalises incorrect predictions by assigning higher loss when the discriminator confidently misclassifies a sample, provides smooth gradients, and aligns with the minimax objective of the generative model, as $\mathcal{G}$ minimises the log-probability that $\mathcal{D}$ correctly identifies its outputs as fake — precisely what BCE captures.

3.2.5 Evaluation Models, Metrics and Setup

To properly test and evaluate the adversarial images generated by the generative model, they were assessed against pre-trained DNN models that achieved high accuracy on the dataset described in Section 3.1. For this purpose, five well-known state-of-the-art models were selected and trained on a multi-class classification task, achieving an average accuracy of $\simeq 96\%$. The models used were AlexNet (Krizhevsky et al., 2012), VGG-16 and VGG-19 (Simonyan et al., 2015), and ResNet18 and ResNet152 (K. He et al., 2016a). The

results of this classification were used to calculate the FR metric proposed by data-free universal approaches (Mopuri et al., 2017; Mopuri et al., 2018). However, the focus was not solely on the percentage of misclassification. Since the models do not achieve 100% accuracy on the original images, each original image was first tested against the classification models. Only if the original image was correctly classified was its adversarial counterpart considered; otherwise, the adversarial image was excluded. This procedure ensures transparent and rigorous scientific results.

Additionally, the FR metric only quantifies misclassification accuracy. To provide a more qualitative evaluation, the FID (Heusel et al., 2017) and LPIPS (R. Zhang et al., 2018) metrics were used to assess the similarity between the generated and original images (see Section 2.6). Human visual inspection was also employed to evaluate the quality of the generated samples.

Finally, regarding the experimental setup, all experiments were done using Python 3.12 with PyTorch 2.6. Manual verification was performed to monitor the training process and analyse loss behaviour, allowing for appropriate adjustments. The study was carried out on a workstation equipped with a single NVIDIA GeForce GTX 1650 GPU (4 GB) using CUDA 12.4.

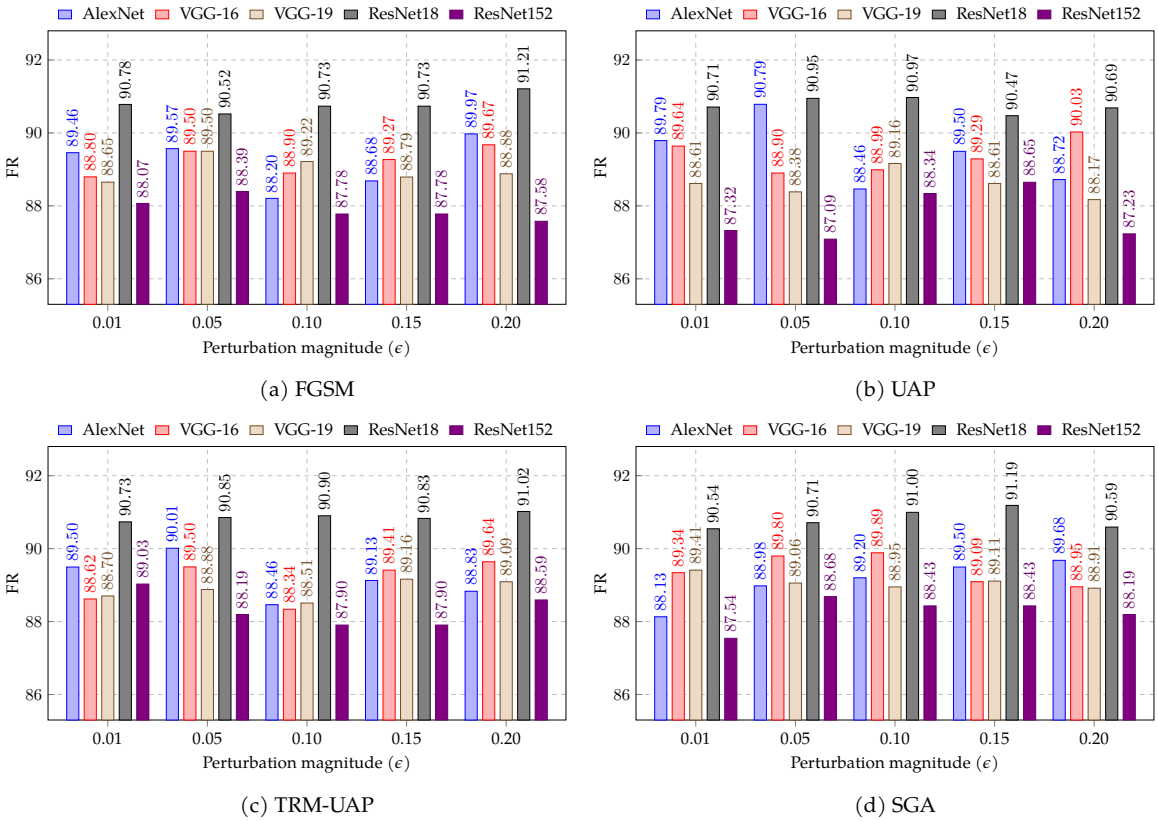
3.3 Experimental Evaluation Results and Analysis

Following the methodology and experimental evaluation environment detailed above, this section presents the results obtained. For clarity, the section is divided into three stages of testing and evaluation, each accompanied by the corresponding results. In the first stage, we present the overall performance of the adversarial generative model according to the evaluation metrics described in Section 3.2.5. Next, we examine the effect of batch size on the training loss of both components of the generative model. Finally, a qualitative assessment of the generated adversarial images is conducted through human visual inspection.

3.3.1 Adversarial DCGAN Performance

As aforementioned, this section presents the DCGAN performance according to the evaluation metrics FR, FID, and LPIPS. Starting with the FR metric, Figure 3.2 shows the results obtained in percentage. The results are presented for each attack — FGSM, UAP, TRM-UAP, and SGA — and evaluated against the classification models for each perturbation magnitude ξ used (see Section 3.2.2).

Regarding the figure mentioned above, the FGSM attack produces consistent results across different perturbation magnitudes ξ and models, with FR values ranging from approximately 86% to 91%. ResNet152 was the most challenging model to fool, with an average FR of 87.92%, while ResNet18 was the easiest, achieving an average FR of 90.79%. VGG-16, VGG-19, and AlexNet showed comparable performance, with minor variations depending on the perturbation magnitude ξ. Similar to the FGSM attack, all three UAP-based attacks produced consistent results, with ResNet152 being the most



(a) FGSM (b) UAP

(c) TRM-UAP (d) SGA

Figure 3.2: *Performance results for the FR metric. Higher FR indicates greater attack effectiveness.*

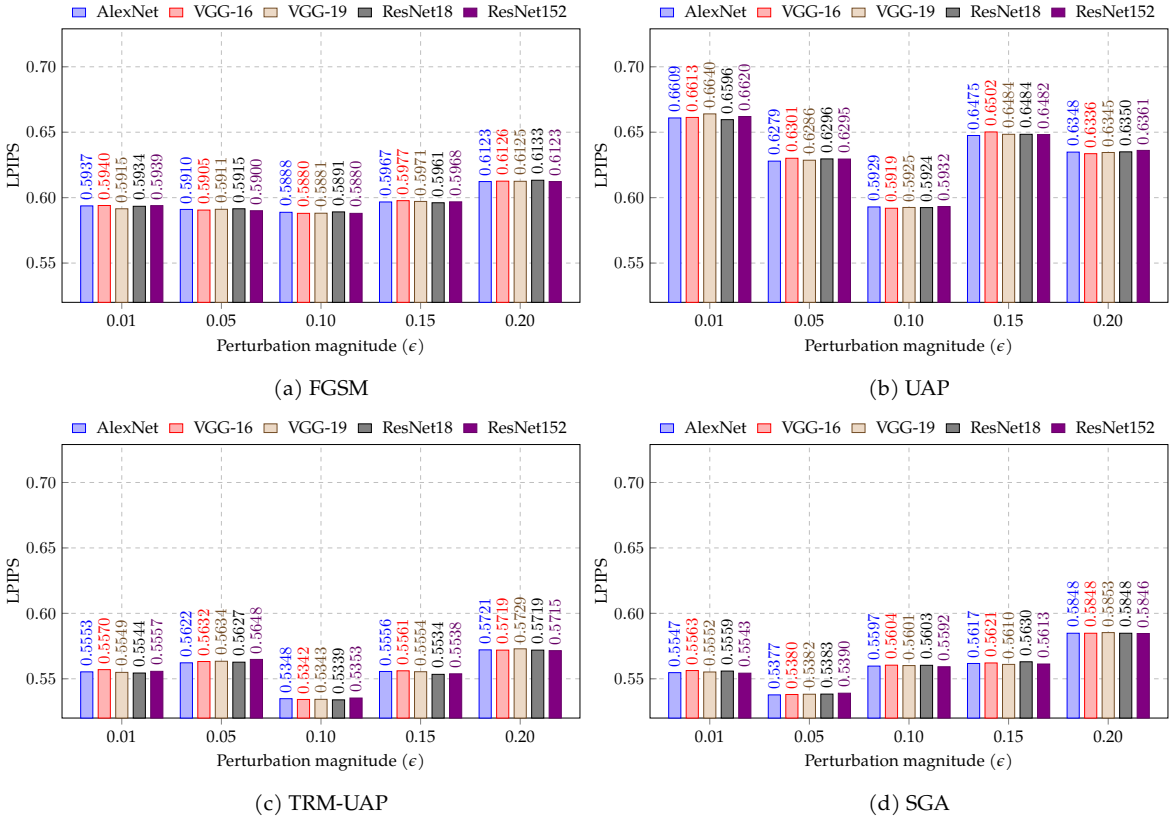
difficult model to fool and ResNet18 the easiest, achieving FR values above 90% across all perturbation magnitudes ξ. Notably, for both the UAP and TRM-UAP attacks, AlexNet achieved strong FR results at $\xi = 0.01$ and $\xi = 0.05$, with an average FR of 90.03%.

To assess the image quality evaluation, we began with the LPIPS. Figure 3.3 shows the results obtained for this metric under the same experimental design described above.

The results indicate that different perturbation magnitudes ξ and attack types produced distinct outcomes. The FGSM attack showed slight variations across the first four perturbation magnitudes but exhibited a clear increase in LPIPS at $\xi = 0.20$, which aligns with theoretical expectations: larger perturbation magnitudes tend to degrade image quality, resulting in higher LPIPS values. The TRM-UAP and SGA attacks achieved the best results, with a slight decrease in LPIPS at perturbation magnitudes $\xi = 0.10$ and $\xi = 0.05$, respectively. In contrast, the UAP attack yielded the poorest results, with an average LPIPS of 0.6334. Notably, the lowest LPIPS score for UAP occurred at $\xi = 0.10$, contradicting theoretical expectations and suggesting potential inconsistencies in the relationship between perturbation magnitude and perceptual similarity for this attack.

Lastly, Figure 3.4 presents the results obtained for the FID metric, based on the same values used in the previous evaluations, *i.e.*, all four attacks applied with the five different perturbation magnitudes across the four classification DNN models.

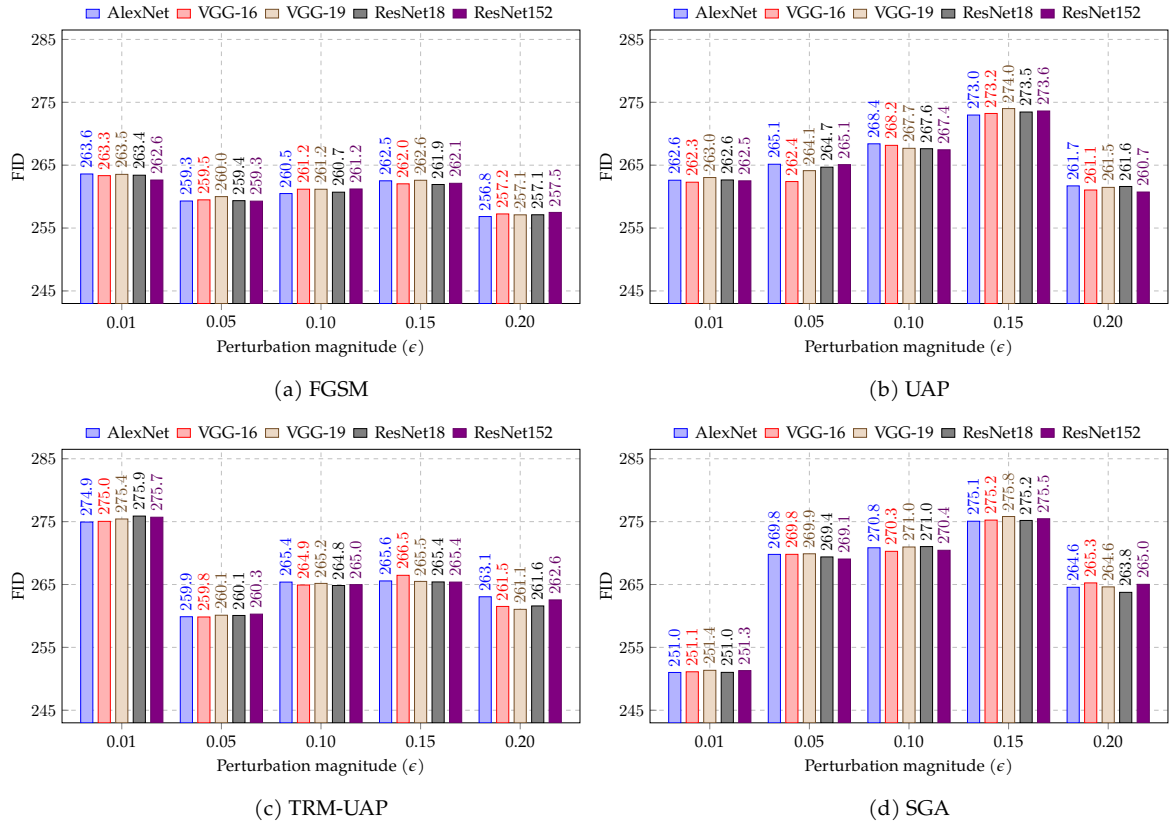
Similar to LPIPS, the FID scores did not vary significantly across models but showed clear differences between perturbation magnitudes. For the FGSM attack, inconsistent results were observed, with the $\xi = 0.20$ magnitude yielding the lowest score, contrary



(a) FGSM (b) UAP

(c) TRM-UAP (d) SGA

Figure 3.3: *Performance results for the LPIPS metric. Lower LPIPS means better image quality.*



(a) FGSM (b) UAP

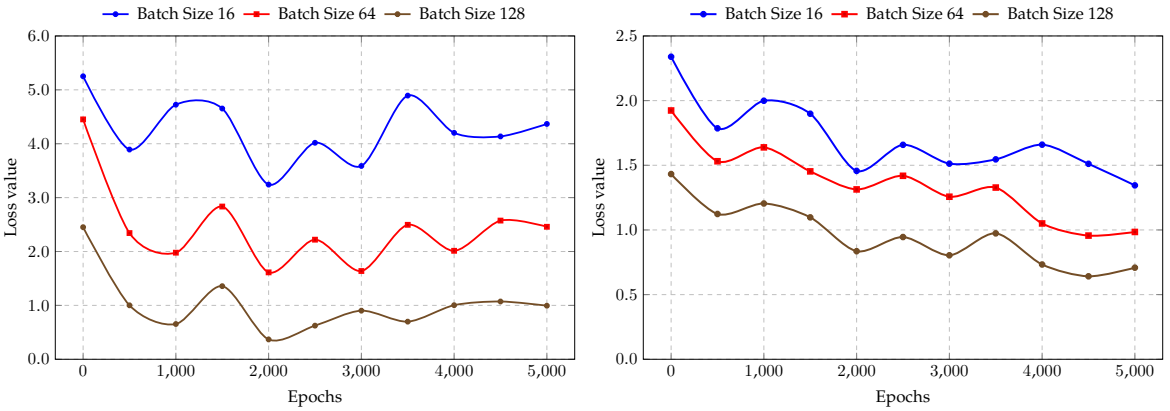
(c) TRM-UAP (d) SGA

Figure 3.4: *Performance results for the FID metric. Lower FID means better image quality.*

to theoretical expectations. In the case of UAP, the metric increased across the first four magnitudes but dropped sharply at the highest one, with an average difference of 12 points. Comparable inconsistencies were noted for TRM-UAP and SGA, where the lowest perturbation produced the highest FID score and the largest the lowest, respectively.

3.3.2 Batch Size Impact on Adversarial Samples

As stated in Section 3.2.4, we conducted a test to determine the optimal batch size for training the generative model. For this purpose, three distinct values were evaluated: 16, 64, and 128. The experiments were carried out sequentially, starting with a batch size of 64, followed by 16, and finally 128. To assess the effect of these configurations, the losses of both components of the generative model, i.e., $\mathcal{G}$ and $\mathcal{D}$, were analysed. As this represented one of the first experiments conducted with this setup, training was extended to a total of 5 000 epochs. The results of this test are shown in Figure 3.5.



(a) Generator loss.

(b) Discriminator loss.

Figure 3.5: *Batch size study showing the loss during both $\mathcal{G}$ and $\mathcal{D}$ training for batch sizes of 16, 64, and 128. The curves for batch size 16, 64, and 128 are represented in blue, red, and brown, respectively.*

The analysis of this study demonstrates that, for both $\mathcal{G}$ and $\mathcal{D}$ training, a larger batch size of 128 produced improved loss metrics. In this particular case of the DCGAN architecture, the optimal loss for both components is characterised by their convergence, *i.e.*, approaching a point of equilibrium. Furthermore, examining the loss curves indicates a more effective training process and ultimately improved image generation by $\mathcal{G}$. Notably, when comparing batch sizes of 16 and 64, the batch size of 64 consistently outperformed 16 for both $\mathcal{G}$ and $\mathcal{D}$, resulting in superior outcomes.

3.3.3 Evaluating Adversarial Image Quality

Upon analysing the extracted results and computing all evaluation metrics, human visual inspection, as described in the methodology, was necessary to assess image quality and fidelity. Notably, while the FR metric indicated strong performance, LPIPS and FID showed less favourable results, highlighting the need for human evaluation. For this purpose, a set of images from the dataset was selected, classified, and their corresponding

adversarial samples were also evaluated.

Figure 3.6: *Comparison of images generated by the DCGAN using the TRM-UAP attack with a magnitude $\xi = 0.10$, targeting the ResNet18 model, alongside real images. The top row displays the original samples with their correct labels. The bottom row shows images generated by the DCGAN, with the classification labels attributed by the model.*

Figure 3.6 presents a comparison between real images and those generated by the model under the TRM-UAP attack with a perturbation magnitude of $\xi = 0.10$ applied to the ResNet18 model, alongside their predicted labels. It is important to note that the labels for the real images correspond to their correct classifications.

3.3.4 Discussion of Results

The results obtained in the previous section fall into three distinct areas of interest: the performance of the generative model in terms of misclassification, measured by the FR metric; the final image quality and fidelity, assessed using the LPIPS and FID metrics together with human-eye observation; and the impact of batch size on the training process of the generative model.

Adversarial DCGAN Performance

Regarding the first area of interest, and focusing solely on the FR metric, we conclude that the adversarial generative model is highly effective at producing adversarial images that mislead all five pre-trained DNN models. The FR ranges from a minimum of 87% to a maximum of 91% across different models. In practical terms, out of 5 000 evaluation images, the models failed to classify correctly, on average, approximately 4 400 images — a substantial result.

Image Quality and Fidelity

Despite the high FR, and as noted above, image quality and fidelity also needed to be evaluated. In this respect, the results were considerably worse. The LPIPS metric scored between 0.53 and 0.66, whereas good similarity should lie in the range of 0.10 to 0.30; values above 0.30 indicate perceptible and significant differences. For the FID metric, the situation was even more severe, with values ranging from 251 to 275. By comparison, ac-

ceptable quality typically falls between 10 and 50, with values between 0 and 10 considered optimal. These results immediately indicated that something was wrong with the generated image quality, which was confirmed through visual inspection. When comparing the original and adversarial images, the differences were substantial: the content of the images was often unrecognisable, with only silhouettes and colours faintly discernible, while overall quality and fidelity were extremely poor.

Batch Size Impact on Adversarial Samples

To determine the most suitable batch size, we not only conducted tests but also analysed its impact on the training procedure of the generative model. The results, presented in the previous section, show that a larger batch size leads to lower loss values for both components of the architecture, $\mathcal{G}$ and $\mathcal{D}$. It is important to note that the optimal point is reached when $\mathcal{G}$ successfully learns the true data distribution, making its outputs indistinguishable from real data. This corresponds approximately to $\mathcal{G}$ achieving a loss of 0.693 and $\mathcal{D}$ a loss of 1.386. These values indicate that $\mathcal{G}$ is generating data of sufficient quality that $\mathcal{D}$ can no longer reliably distinguish it from real data, representing the theoretical balance between the two models. From the analysis, a batch size of 128 provides the best performance in reaching this equilibrium, followed by 64, with 16 performing the worst, indicating that larger batch sizes are more effective for this study.

3.4 Chapter Summary

As mentioned at the beginning of this chapter, the objective of this study was to determine whether a generative model, specifically a DCGAN, can learn the adversarial traits of adversarial images and subsequently generate adversarial images on demand that mislead a given DNN model. To this end, a pre-established DCGAN was trained on datasets corresponding to different adversarial attacks and perturbation magnitudes ξ.

Against five pre-trained DNN models, the evaluation considered both the misclassification rate of the generated images, measured using FR, and their quality and fidelity, assessed using the LPIPS and FID metrics in combination with human-eye observation. Additional tests were conducted to determine the optimal batch size for effective training. The results were striking: FR values ranged from 87% to 91%, demonstrating strong adversarial performance. However, image quality was severely compromised, with both LPIPS and FID performing poorly. Human observation confirmed that the generated images bore little resemblance to the originals, capturing only silhouettes and colours.

Given these results, it was necessary to ensure that adversarial images preserved high quality and fidelity while achieving a high FR. To address this, we proposed a multi-objective generative model using multiple discriminators that compete on two fronts: image quality and resemblance to the original image, as well as capturing adversarial traits. This study was carried out and is fully described in the next chapter, Chapter 4, titled Multi-Objective Generative Model for Adversarial Sample Generation.

4

Multi-Objective Generative Model for Adversarial Sample Generation

The purpose of this chapter is to present the work developed as a continuation of the previous chapter (see Chapter 3). The conclusions of that study pointed to the need for a multi-objective generative model to generate adversarial images. This chapter details the development of such a model. The study aimed to demonstrate that a multi-objective model — *i.e.*, a model with multiple discriminators — can be trained to simultaneously address the image quality issues observed in the previous study and accurately capture the adversarial traits of adversarial images. To ensure the robustness of the model, state-of-the-art defences from the literature were included in the evaluation to assess both the quality, fidelity and robustness of the images generated. Building on this idea, the chapter presents the proposed generative architecture developed in this study, followed by the adapted methodology from the previous study. The experimental evaluation environment and results are presented, followed by a discussion of the findings and tests conducted, and concluded with a brief summary of the chapter. The contents of this chapter were presented at the *2025 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining UCM Track — KDD-UMC* (Areia et al., 2025a) and published in the *Social Science Research Network — Elsevier* (Areia et al., 2025b). Additionally, this work is publicly available in the official GitHub repository[1].

4.1 Design of the Proposed Generative Architecture

To fulfil the objectives of this study, and since no architecture had been specifically designed for this aspect of research, we developed a novel multi-objective architecture, termed MOSA-GAN, based on the most recent generative model, SuperstarGAN[2] (Ko et al., 2023).

[1] GitHub repository: `https://github.com/ipleiria-ciic/multiobjective-adversarial-gan`

[2] It should be emphasised that the original model does not incorporate multi-objective functionality. The choice of this architecture was motivated by its strong benchmark performance, particularly in terms of image quality and fidelity.

Given this context, the proposed multi-objective adversarial generative architecture, MOSA-GAN, illustrated in Figure 4.1, comprises two discriminators, denoted as $\mathcal{D}$ and $\mathcal{D}'$, a generator $\mathcal{G}$, a classifier C, and an encoder $\mathcal{E}$. The discriminator $\mathcal{D}$ operates as a standard, or *vanilla*, discriminator, determining whether the input image is real or synthetic, thereby guiding $\mathcal{G}$ to produce outputs that closely approximate the originals. By contrast, the discriminator $\mathcal{D}'$ is designed to extract adversarial features from perturbed images and to promote their subtle incorporation into the generated outputs, while ensuring they remain imperceptible to the human eye.

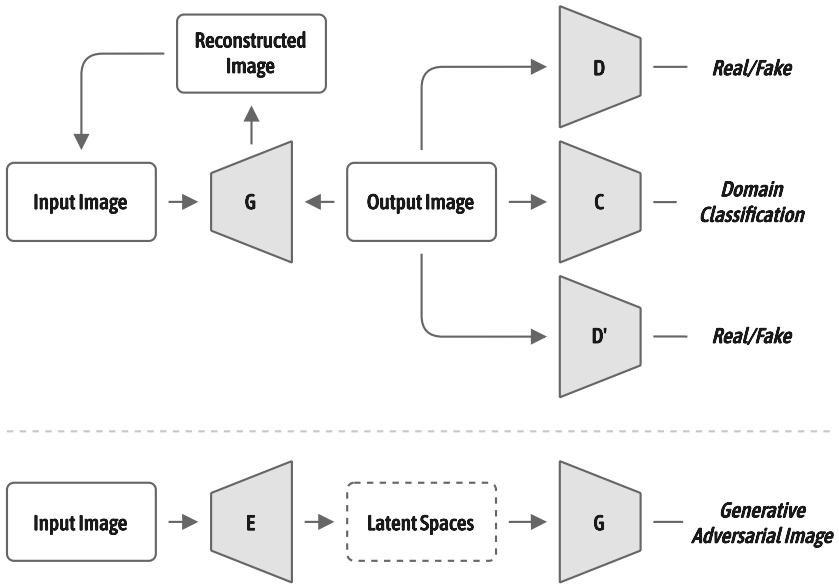


Figure 4.1: *Proposed MOSA-GAN architecture. The upper diagram represents the multi-objective generative framework with multiple discriminators, while the lower diagram illustrates the encoder training process, designed to reconstruct real images by leveraging the trained generator.*

Still regarding the MOSA-GAN architecture, the generator $\mathcal{G}$ employs an encoder-decoder framework with integrated instance normalisation. It begins with a single convolutional layer followed by a ReLU activation, and is followed by two downsampling blocks, each comprising a convolutional layer, instance normalisation, and ReLU activation. The network's core consists of six residual blocks, each containing two convolutional layers with instance normalisation and connected via skip connections to support information flow and maintain gradient stability. After the residual blocks, two upsampling blocks are implemented using transposed convolutions, each coupled with instance normalisation and ReLU activation. The final output layer uses a Tanh activation function to generate the output image.

Turning to the discriminator $\mathcal{D}$, it employs a PatchGAN (Demir et al., 2018) architecture, consisting of six convolutional layers with spectral normalisation and LeakyReLU activations. The adversarial discriminator $\mathcal{D}'$ uses a similar convolutional backbone but ends with a fully connected, spectral normalised linear layer for classification. The auxiliary classifier C mirrors the convolutional structure of the discriminators and concludes with a global convolutional classification head, adapted to the downsampled spatial dimensions of the input image.

Aligned with the objectives outlined in Chapter 3 — to take an image as input and produce the same image with a specified perturbation ξ — it is necessary to capture its latent space information. To achieve this, we employ an encoder $\mathcal{E}$ that learns structured latent representations of real data, aiding $\mathcal{G}$ in generating more realistic and diverse outputs. Consequently, rather than using $y = \mathcal{G}(z)$ to produce an adversarial image, we use $y = \mathcal{G}(\mathcal{E}(x))$, where $\mathcal{E}(x)$ represents the output of $\mathcal{E}$ when the real image x is input.

Additionally, and as the key feature of the MOSA-GAN architecture, it is possible to assign different weights (*importance*) to each discriminator. This allows control over the objectives of the study. To prioritise a cleaner and higher-fidelity image, a greater weight is assigned to the vanilla discriminator $\mathcal{D}$; conversely, to emphasise perturbations, a higher weight is given to the adversarial discriminator $\mathcal{D}'$. To implement this, we applied the concept of nadir slack. The nadir slack represents the difference between the nadir point and a reference point, referred to as the *objective point*. Experimentally, the slack value is used to dynamically update the nadir point during training, improving stability (Albuquerque et al., 2019), as described by the following equation:

$$z^{nadir} = \max_i \ell_i \cdot \varsigma^{nadir} + \kappa, \tag{4.1}$$

where ℓ_i denotes the individual objective losses, and κ is a small constant set to $\kappa = 1\text{e}{-8}$. This nadir point, z^{nadir}, is subsequently used during training, together with the losses from the various objectives, to compute the HV. The resulting HV is then employed to calculate the loss of $\mathcal{G}$, providing a measure of how effectively the training process explores the objective space towards the ideal point of this study, which is within the context of multi-objective optimisation.

4.2 Methodology

The methodology of this study is largely consistent with that of the previous work (see Section 3.1). However, in this case, point three — *Generative Model Selection and Training* — has been revised. In addition, a new sixth point has been introduced, addressing the robustness assessment against state-of-the-art defence mechanisms reported in the literature. Consequently, the methodology is as follows:

1. **Data Acquisition and Preprocessing:** Follows the same process as in Section 3.1.
2. **Adversarial Attack Synthesis:** Follows the same process as in Section 3.1.
3. **MOSA-GAN Model Training:** Training of the MOSA-GAN model to capture both adversarial and original characteristics from the image set, and produce synthetic adversarial samples on-demand.
4. **Representation Encoder Training:** Follows the same process as in Section 3.1.
5. **Performance Testing and Evaluation:** Follows the same process as in Section 3.1.
6. **Robustness Assessment Against Defence Mechanisms:** Comprehensive assessment of the robustness of the trained model against state-of-the-art defence mechanisms reported in the literature.

MOSA-GAN Model Training

At this stage, unlike the previous study, the generative architecture had already been selected: the MOSA-GAN model proposed in the previous section (see Section 4.1). This architecture is expected to satisfy the requirements set out in the previous methodology, namely: to produce high-quality images, ensure stable training, and support effective hierarchical feature learning, while also capturing the adversarial traits of an input image in an effective and imperceptible manner.

Robustness Assessment Against Defence Mechanisms

In this stage, we test the proposed generative model against state-of-the-art defences reported in the literature. The objective is to determine whether adversarial images generated by MOSA-GAN, when processed by a given defence, can reduce the misclassification rate (FR) of pre-trained DNN evaluation models. If the reduction in FR is smaller than that reported in the literature for the corresponding defence, we conclude that the images generated by MOSA-GAN are, in fact, strong adversarial examples.

Main Algorithm

For greater clarity regarding the methodology applied in this study, Algorithm 1 is presented. It provides a structured overview of the adversarial training process of MOSA-GAN, including the evaluation procedure for the generated images.

Algorithm 1: *Adversarial training process of MOSA-GAN with an evaluation procedure.*

Require: Surrogate DNN models $f = \{f_1, \ldots, f_5\}$ for evaluation
Input: Original dataset $\mathbb{D}$ and a list of attacks $\mathbb{I}$
Output: JSON file $m = \{m_1, \ldots, m_4\}$
  for $i \in I$, where $\mathbb{I} = \{\text{FGSM}, \text{UAP}, \text{TRM-UAP}, \text{SGA}\}$ **do**
    Generate perturbation δ_i
    Create perturbed dataset: $\mathbb{D}_i = \mathbb{D} + \delta_i$
    Train generator $\mathcal{G}_i$ with objectives:
      $\mathcal{D}(\mathcal{G}(x)) \approx \mathcal{D}(x)$ ▷ Preserve the realism of the original image.
      $\mathcal{D}'(\mathcal{G}(x)) \approx \mathcal{D}(x_i)$ ▷ Capture adversarial features of the malicious image.
    Train encoder $\mathcal{E}$ on $\mathbb{D}$ to extract latent codes z:
      $z = \mathcal{G}(\mathcal{E}(x)),\ x \in \mathbb{D}$
    for all $x \in \mathbb{D}$ **do**
      if $\hat{y} = f(x) = y$ by all $f_j \in f$ **then**
        for all $f_j \in f$ **do**
          Compute $\text{FR}(f_j, \mathbb{D}_i)$
        end for
      end if
    end for
    Compute $\text{FID}(\mathbb{D}, \mathbb{D}_i)$
    Compute $\text{LPIPS}(\mathbb{D}, \mathbb{D}_i)$
    $m_i = \{\text{FR}_i,\ \text{LPIPS}_i,\ \text{FID}_i\}$
  end for
  return $m = \{m_1, \ldots, m_4\}$

4.3 Experimental Evaluation Environment

This section describes the experimental environment of this study, including the rationale behind each component to ensure alignment with the stages of the methodology outlined in the preceding section. It is important to note that, since part of the methodology is inherited from the previous study, there is no need to repeat the explanation of the corresponding experimental environment. For components where the methodology remains unchanged, please refer to the relevant sections of the previous study: *Data Collection and Preprocessing* (Section 3.2.1), *Adversarial Attack Synthesis* (Section 3.2.2), and *Evaluation Models, Metrics and Setup* (Section 3.2.5).

4.3.1 Network Architecture

The network architecture was presented at a high level in Section 4.1; however, this section provides a more detailed description of the settings, layers, and related configurations. As previously stated, MOSA-GAN is partially based on the state-of-the-art GAN, SuperstarGAN (Ko et al., 2023), with several key modifications. SuperstarGAN consists of a generator $\mathcal{G}$, a discriminator $\mathcal{D}$, and a classifier C. Since the objective is to generate images that resemble the originals while incorporating adversarial traits, we introduced a second adversarial discriminator, $\mathcal{D}'$, to extract adversarial features during training, along with a custom encoder, $\mathcal{E}$, to learn latent representations of the real data.

Generator

The generator $\mathcal{G}$ starts with a 7×7 convolutional layer, followed by downsampling through two 4×4 convolutions, each doubling the number of feature maps. The bottleneck comprises six residual blocks, and upsampling is performed using two transposed convolutions, reducing the feature maps to produce a 3-channel image with a Tanh activation. The input consists of an image concatenated with domain label information. Listing 3 presents the Python class implementation of the generator $\mathcal{G}$.

Vanilla and Adversarial Discriminator

The vanilla discriminator $\mathcal{D}$ consists of multiple convolutional layers with spectral normalisation, starting with a 4×4 convolution followed by a LeakyReLU activation. It includes six convolutional layers, each doubling the number of feature maps, and concludes with a 3×3 convolution that outputs a real-or-fake score. The adversarial discriminator $\mathcal{D}'$, designed for adversarial training, follows a similar structure to $\mathcal{D}$ but ends with a fully connected layer that classifies the input into one of $n = 10$ labels. Listing 1 and Listing 2 present the Python class implementations of the vanilla discriminator $\mathcal{D}$ and the adversarial discriminator $\mathcal{D}'$, respectively.

Classifier and Encoder

The classifier C consists of convolutional layers with LeakyReLU activations, beginning with a 4×4 convolution and followed by six downsampling layers. The final layer outputs a class-dimensional vector representing the class scores. The encoder $\mathcal{E}$ consists of

convolutional layers with ReLU activations, starting with a 4×4 convolution and applying six downsampling layers. A final convolution with an adaptive kernel size produces a class-dimensional vector representing the inferred domain vector. Listing 4 and Listing 5 present the Python class implementations of the classifier C and the encoder $\mathcal{E}$, respectively.

4.3.2 Hyperparameter Selection

In this stage of the work, we conducted a wide range of experiments, as in the previous study, where this stage was also the most time-consuming. Defining the optimal values for each parameter in the architecture training process is highly challenging. Nevertheless, this process was carried out with caution and attention to ensure rigorous scientific work and methodological consistency. Table 4.1 presents the final values of each parameter used in this study. The rationale behind each choice will be explained later.

Table 4.1: *The final hyperparameters and their values used in the MOSA-GAN training process.*

Hyperparameter	**Final Setting**
Batch size	128
Input image resolution	128×128
Input channels	3
Training epochs	100 000
Convolutional filters in generator $\mathcal{G}$	32
Convolutional filters in discriminators $\mathcal{D}$ and $\mathcal{D}'$	32
Convolutional filters in classifier C	32
Convolutional filters in encoder $\mathcal{E}$	32
Residual blocks in generator $\mathcal{G}$	6
Residual blocks in discriminators $\mathcal{D}$ and $\mathcal{D}'$	6
Residual blocks in classifier C	6
Residual blocks in encoder $\mathcal{E}$	6
Weight for domain classification loss	0.25
Weight for reconstruction loss	1.3
Weight for gradient penalty	1.0
Nadir slack ς^{nadir}	1.1
Weights for the discriminators $\mathcal{D}$ and $\mathcal{D}'$	$[0.65, 0.35]$
Adam learning rate (ℓ_r) for generator $\mathcal{G}$	$1.0\text{e}{-4}$
Adam learning rate (ℓ_r) for discriminators $\mathcal{D}$ and $\mathcal{D}'$	$1.0\text{e}{-4}$
Adam learning rate (ℓ_r) for classifier C	$1.2\text{e}{-4}$
Adam β_1	0.0
Adam β_2	0.999
ReduceLROnPlateau reduction factor (r_d)	0.8
ReduceLROnPlateau patience (p)	30
ReduceLROnPlateau minimum learning rate $(\min \ell_r)$	$1.0\text{e}{-10}$

All of these hyperparameters were carefully selected rather than arbitrarily defined. It is important to note that the testing scenario was based on the FGSM attack dataset with a perturbation magnitude of $\xi = 0.10$, with the aid of the optimisation tool mentioned later to automatically determine the best values.

Regarding the batch size, this was chosen based on the empirical experimentation conducted in the previous study (see Section 3.3.2), and thus the same value was retained. Additionally, since the MOSA-GAN model inherits from SuperstarGAN (Ko et al., 2023), several hyperparameters were also preserved, as they had already been shown to yield optimal results in that configuration. These include the input image resolution, the convolutional filters used across all components, and the residual blocks incorporated in each component.

Similar to the previous study, we employed the Optuna (Akiba et al., 2019) optimisation tool in this work to determine the best possible settings for the remaining parameters. The parameters submitted to this tool included the weights for the reconstruction loss, the gradient penalty, and the domain classification loss, as well as the nadir slack. Additionally, the learning rates ℓ_r and the Adam (Diederik P. Kingma et al., 2017) optimiser betas, β_1 and β_2, were tuned for all components. The reduction factor r_d, patience p, and minimum learning rate min ℓ_r for the ReduceLROnPlateau scheduler (Paszke et al., 2019) were also optimised.

Regarding the weights for the domain classification loss, reconstruction loss, and gradient penalty, four additional candidate values were considered around the final value: two positive increments and two negative increments, each with a step of 0.5. After Optuna optimisation, the middle value was selected for each parameter, corresponding to the final values presented in the table above. The nadir slack, as suggested by Albuquerque et al. (2019), should lie within the following range $[1.05, 1.1]$. To determine the optimal value, we tested the search space $\varsigma^{nadir} = \{1.05, 1.06, 1.07, 1.08, 1.09, 1.10\}$, and the highest value, $\varsigma^{nadir} = 1.10$, was selected.

For the Adam optimiser, five learning rates, ℓ_r, were tested for all components using the same search space: $\ell_r = \{1.0\text{e}{-4}, 1.1\text{e}{-4}, 1.2\text{e}{-4}, 1.3\text{e}{-4}, 1.4\text{e}{-4}\}$. For the generator $\mathcal{G}$, both discriminators $\mathcal{D}$ and $\mathcal{D}'$, and for the encoder $\mathcal{E}$, the final learning rate was set to $\ell_r = 1.0\text{e}{-4}$, while for the classifier C it was set to $\ell_r = 1.2\text{e}{-4}$. The Adam betas were also optimised within a specific defined search space, yielding final values of $\beta_1 = 0.0$ and $\beta_2 = 0.999$.

For the ReduceLROnPlateau scheduler, the search spaces were maintained similar to those defined in the previous study: $r_d = \{0.5, 0.6, 0.7, 0.8, 0.9\}$, $p = \{10, 20, 30, 40, 50\}$, and $min\ \ell_r = \{1\text{e}{-4}, 1\text{e}{-5}, 1\text{e}{-6}\}$. After optimisation, the best values obtained were $r_d = 0.8$, $p = 30$, and $min\ \ell_r = 1\text{e}{-6}$. Additionally, following the methodology of the previous study, it is important to note that BCE was used as the canonical loss function, consistent with the original work of Ian J. Goodfellow et al. (2014).

Regarding the number of epochs, and as in the previous study, an assessment was conducted to determine the appropriate value. Specifically, the loss values of $\mathcal{G}$, $\mathcal{D}$, $\mathcal{D}'$, and C were monitored and evaluated under the testing settings described at the beginning of this section. It was observed that after approximately epoch 95,000, the training process became unstable, with the loss of $\mathcal{G}$ diverging. Various tests and hyperparameter adjustments were attempted to mitigate this issue; however, no improvements were achieved.

Finally, to conclude the hyperparameter selection section, it is important to note that extensive work and an ablation study were conducted to determine the optimal weights for both discriminators, $\mathcal{D}$ and $\mathcal{D}'$. This study will not be detailed here, as Section 4.4.2 is entirely dedicated to it.

4.3.3 Defences Implemented

As outlined in Section 4.2, this study also aims to evaluate the robustness of the proposed MOSA-GAN model against state-of-the-art defences reported in the literature. For this purpose, we focus on two well-known and easily applicable defences: adversarial training (Ian J Goodfellow et al., 2015) and input transformation (Tian et al., 2024).

Adversarial Training

In adversarial training, adversarial images generated by the MOSA-GAN model are used to retrain the evaluation models, which were originally trained on clean images in the previous study. This ensures that the DNN models can learn the adversarial traits of these images and recognise them during classification tasks, hence improving robustness against attacks.

Input Transformation

For input transformation, we follow the approach of Tian et al. (2024), applying different types of blur to the adversarial images before classification. This aims to obscure the adversarial traits and reduce the misclassification rate — FR — of the models. Four types of blur are applied, consistent with the original study: Gaussian blur, Median blur, Bilateral blur, and Affine blur.

4.3.4 Experimental Setup

All experiments were conducted using Python 3.12 with PyTorch 2.6. Manual verification was employed to monitor the training process and analyse loss behaviour. The study was performed on a workstation equipped with a single NVIDIA GeForce RTX 4080 Super GPU (16 GB) using CUDA 12.4.

4.4 Experimental Evaluation Results and Analysis

Following the methodology and experimental evaluation environment described above, this section presents the results obtained. It is divided into four stages of testing and evaluation, each accompanied by the corresponding results. In the first stage, we present the overall performance of the MOSA-GAN model according to the evaluation procedure outlined earlier. Next, we present the ablation study assessing the influence of the discriminator weights on the model's general performance. This is followed by the results obtained against state-of-the-art defences (see Section 4.3.3). Finally, we provide a qualitative assessment of image quality and discuss the overall results achieved in this study.

4.4.1 MOSA-GAN Model Performance

This section presents the results obtained for the MOSA-GAN model according to the three evaluation metrics — FR, FID, and LPIPS. The results are summarised in Table 4.2, where they are reported separately for each evaluation metric across the different DNN classifier models. The values shown correspond to the averages over all perturbation magnitudes ξ used (see Section 3.2.2), along with the respective standard deviations.

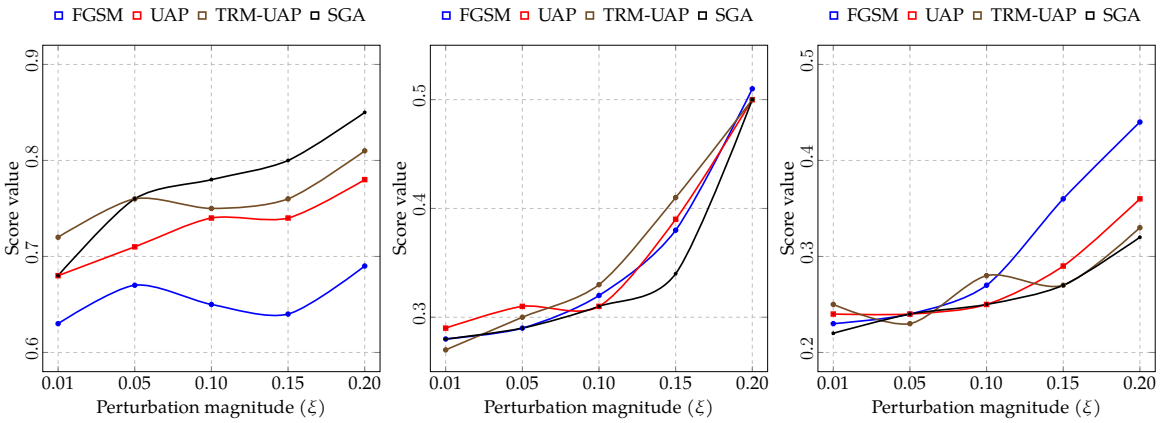
Table 4.2: *Summary of evaluation metrics for MOSA-GAN performance analysis.*

Attack	AlexNet	ResNet152	ResNet18	VGG-16	VGG19
Fooling Rate (FR)					
FGSM	74.5±1.5	80.3±4.7	81.1±3.6	72.0±2.6	70.4±2.6
UAP	70.3±4.2	66.0±3.1	81.6±3.4	72.1±4.7	75.3±4.8
TRM-UAP	76.3±5.0	82.9±2.9	84.2±4.0	74.5±6.5	73.3±6.4
SGA	78.3±9.0	77.9±5.9	77.6±5.7	77.7±7.1	75.5±6.0
Fréchet Inception Distance (FID)					
FGSM	36.6±12.0	33.7±8.2	36.1±10.1	36.4±7.6	35.9±10.9
UAP	36.9±8.5	33.9±6.4	37.2±10.8	35.4±9.6	37.2±9.8
TRM-UAP	36.5±8.1	34.6±7.8	36.7±11.4	37.4±9.9	36.5±10.8
SGA	32.6±7.8	34.9±10.1	36.4±8.8	34.5±10.4	34.0±9.0
Learned Perceptual Image Patch Similarity (LPIPS)					
FGSM	0.30±0.09	0.29±0.10	0.30±0.10	0.31±0.09	0.31±0.09
UAP	0.27±0.06	0.27±0.06	0.26±0.06	0.26±0.06	0.26±0.06
TRM-UAP	0.25±0.05	0.26±0.06	0.25±0.04	0.26±0.05	0.25±0.05
SGA	0.25±0.05	0.24±0.05	0.24±0.05	0.25±0.05	0.25±0.05

Regarding the FR, the overall results range from $\simeq$ 66.0% to 84.2%. ResNet18 achieved the highest average FR at 80.7%, while VGG19 showed the lowest performance, with an average FR of 73.3%. Although some variation in attack transferability was observed across models, the SGA attack demonstrated notable consistency, with an average fluctuation of only 1%. By contrast, other attacks exhibited slightly larger variations but achieved results $\simeq$ 5% higher than those of SGA.

Across all tests, the average FID scores are broadly consistent across attacks and models, ranging from 32.6 to 37.4. Among these, SGA achieves the lowest — and therefore *best* — average score of 34.4, followed by FGSM at 35.7, UAP at 36.1, and TRM-UAP at 36.3. Similarly, the LPIPS scores exhibit limited variation across attacks and methods, ranging from 0.24 to 0.31. Once again, SGA achieves the best performance with the lowest average of 0.24, followed by TRM-UAP at 0.25, UAP at 0.26, and FGSM, which records the highest — and therefore *worst* — score of 0.30.

Since the perturbation magnitudes ξ were averaged in the results presented in the table above, Figure 4.2 is provided to illustrate the correlation between the average evaluation metrics across all tested models and attacks with respect to the different perturbation magnitudes ξ assessed. It is important to note that the FID scores were normalised to the $[0, 1]$ range by dividing the average by 100.



(a) FR (b) FID (c) LPIPS

Figure 4.2: *Correlation between the average evaluation metrics — FR, FID, and LPIPS — and the perturbation magnitude of the generated images for each attack, across the tested models: AlexNet, VGG-16, VGG-19, ResNet18, and ResNet152. FID values were normalised to* $[0,1]$ *by dividing by 100.*

Examining the aforementioned figure and its corresponding results, all attacks display a consistent trend: as the perturbation magnitude ξ increases, the FR generally rises. A minor exception is observed with FGSM at $\xi = 0.10$ and $\xi = 0.15$, where the FR slightly decreases compared to $\xi = 0.05$. In parallel, both FID and LPIPS values increase with larger perturbation magnitudes, reflecting a decline in image quality — an expected outcome, as stronger perturbations naturally degrade visual fidelity.

4.4.2 Influence of Discriminator Weight Variation

As outlined in the hyperparameter selection section (see Section 4.3.2), this study includes an ablation test on the weights assigned to the vanilla and adversarial discriminators — $\mathcal{D}$ and $\mathcal{D}'$ — within the proposed architecture. To conduct this empirical analysis, variations of the discriminator weight configurations were created, each scenario assigned a descriptive name to facilitate interpretation in the forthcoming result plots. The mapping between scenario names and their corresponding weight assignments is presented in Table 4.3. It is important to emphasise that the weights adopted in the final architecture are regarded as the optimal configuration. Consequently, the configuration with weights $\lambda_D = 0.65$ and $\lambda_{D'} = 0.35$ is designated as the final setup and is directly associated with the name of the architecture, *i.e.* MOSA-GAN.

Table 4.3: *Discriminator weight variations used in the ablation test.*

Configuration Name	Weight Values
MOSA-GAN	$\lambda_D = 0.65$ and $\lambda_{D'} = 0.35$
SuperstarGAN Vanilla (Van)	$\lambda_D = 1.00$ and $\lambda_{D'} = 0.00$
SuperstarGAN Adversarial (Adv)	$\lambda_D = 0.00$ and $\lambda_{D'} = 1.00$
SuperstarGAN Vanilla-Weighted (Van_w)	$\lambda_D = 0.75$ and $\lambda_{D'} = 0.25$
SuperstarGAN Adversarial-Weighted (Adv_w)	$\lambda_D = 0.25$ and $\lambda_{D'} = 0.75$
SuperstarGAN Adversarial-Weighted (Adv_w)	$\lambda_D = 0.25$ and $\lambda_{D'} = 0.75$

With the evaluation environment and test scenarios established, we proceeded with the experiments. Table 4.4 presents the results of this study, showing the average FR achieved for each attack across the different setups relative to the DNN models, along with the corresponding standard deviations. For comparison, this study includes the results from the previous study presented in Chapter 3, referred to as *Adversarial DCGAN* (Areia et al., 2025c), which does not employ multi-objective optimisation.

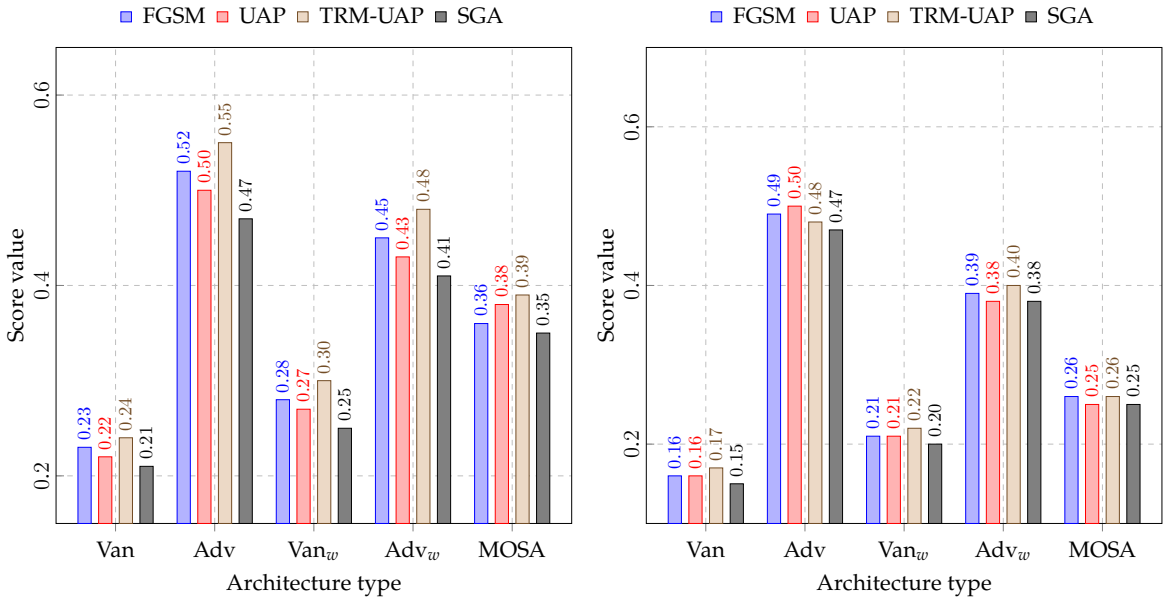
Table 4.4: *Effect of discriminator weight variation on model performance across the different attack types — FGSM, UAP, TRM-UAP and SGA — measured using the FR metric.*

Attack	AlexNet	ResNet152	ResNet18	VGG-16	VGG19
Adversarial DCGAN (Areia et al., 2025c)					
FGSM	86.7±1.4	90.6±4.2	94.4±3.8	83.9±2.4	82.0±2.5
UAP	81.8±4.1	76.8±3.0	94.3±3.3	83.8±4.6	87.7±4.7
TRM-UAP	88.6±4.8	89.7±2.7	95.9±4.1	86.7±6.4	85.4±6.2
SGA	91.4±8.7	90.7±5.8	90.4±5.6	90.5±7.0	87.9±5.9
SuperstarGAN Vanilla ($\lambda_D = 1.0$; $\lambda_{D'} = 0.0$)					
FGSM	32.8±1.4	35.3±4.6	35.7±3.4	31.7±2.4	30.9±2.3
UAP	30.9±3.8	29.0±3.0	35.9±3.5	31.7±4.2	33.1±4.3
TRM-UAP	33.6±4.7	36.5±2.6	37.0±3.9	32.8±6.2	32.3±6.0
SGA	34.5±8.7	34.3±5.5	37.1±5.4	34.2±6.9	33.2±5.8
SuperstarGAN Adversarial ($\lambda_D = 0.0$; $\lambda_{D'} = 1.0$)					
FGSM	86.1±1.3	89.0±4.3	90.1±3.7	87.9±2.4	86.0±2.5
UAP	85.8±4.0	80.5±2.9	90.1±3.2	87.9±4.5	89.0±4.6
TRM-UAP	88.1±4.6	87.0±2.6	89.9±4.2	91.0±6.2	86.5±6.3
SGA	90.6±8.6	91.2±5.5	90.8±5.3	90.1±6.8	89.2±5.9
SuperstarGAN Vanilla-Weighted ($\lambda_D = 0.75$; $\lambda_{D'} = 0.25$)					
FGSM	60.3±1.4	65.1±4.6	65.9±3.5	58.3±2.5	57.0±2.6
UAP	56.9±4.0	53.5±3.0	66.1±3.3	58.4±4.4	61.2±4.5
TRM-UAP	61.8±4.8	67.1±2.7	68.4±4.1	60.3±6.1	59.4±6.2
SGA	63.4±8.8	63.1±5.6	62.8±5.5	62.9±7.0	61.2±5.8
SuperstarGAN Adversarial-Weighted ($\lambda_D = 0.25$; $\lambda_{D'} = 0.75$)					
FGSM	80.4±1.4	86.7±4.3	87.6±3.4	77.7±2.5	76.0±2.5
UAP	75.9±4.0	71.3±3.0	88.1±3.2	77.8±4.3	81.3±4.6
TRM-UAP	82.4±4.7	89.5±2.7	91.0±4.2	80.6±6.3	79.2±6.1
SGA	84.6±8.6	84.2±5.5	83.8±5.4	83.9±6.8	81.5±5.9
MOSA-GAN ($\lambda_D = 0.65$; $\lambda_{D'} = 0.35$)					
FGSM	74.5±1.5	80.3±4.7	81.1±3.6	72.0±2.6	70.4±2.6
UAP	70.3±4.2	66.0±3.1	81.6±3.4	72.1±4.7	75.3±4.8
TRM-UAP	76.3±5.0	82.9±2.9	84.2±4.0	74.5±6.5	73.3±6.4
SGA	78.3±9.0	77.9±5.9	77.6±5.7	77.7±7.1	75.5±6.0

Regarding the results, the Adversarial DCGAN achieved the highest score among the tested alternatives, with an average FR of 88.4%. As expected, the lowest score was obtained by SuperstarGAN Vanilla, with an average FR of 33.9%, while SuperstarGAN Ad-

versarial achieved the second-highest score at 87.4%. SuperstarGAN Vanilla-Weighted and SuperstarGAN Adversarial-Weighted recorded average FR values of 61.7% and 80.1%, respectively. Notably, our proposed architecture, MOSA-GAN, using the baseline configuration described earlier, achieved an average FR of 74.4%.

Alongside the FR results, image quality and fidelity were also evaluated in this stage of testing and evaluation. Figure 4.3 presents the FID and LPIPS values obtained for each tested method, averaged across the four attacks, perturbation magnitudes ξ, and target models. Two points should be noted: for this analysis, the Adversarial DCGAN was excluded, as only the multi-objective variants were considered, and the FID results were normalised to the $[0, 1]$ interval by dividing the original values by 100.



(a) FID (b) LPIPS

Figure 4.3: *Correlation between average evaluation metrics (FID and LPIPS) and weighted network architecture used for sample generation. Results are averaged across the tested models: AlexNet, VGG-16, VGG-19, ResNet18, and ResNet152. FID values were normalised to the [0,1] range by dividing by 100. The column MOSA refers to the default baseline configuration used in this study — MOSA-GAN.*

The results indicate that higher FR values are generally associated with increased — and therefore *worse* — image fidelity metrics. For example, SuperstarGAN Vanilla and SuperstarGAN Vanilla-Weighted exhibit lower FR alongside lower FID and LPIPS scores, suggesting that the generated images closely resemble the originals. In contrast, SuperstarGAN Adversarial and SuperstarGAN Adversarial-Weighted achieve high FR but poor FID and LPIPS scores, reflecting visibly degraded image quality with noticeable perturbations. Our method, MOSA-GAN, achieved a balanced average result, representing a middle ground between the two extremes.

4.4.3 Robustness of MOSA-GAN Against Defensive Techniques

As stated in Section 4.3.3, state-of-the-art defences were tested to evaluate the robustness of the proposed MOSA-GAN model. The defences selected for this study were adversar-

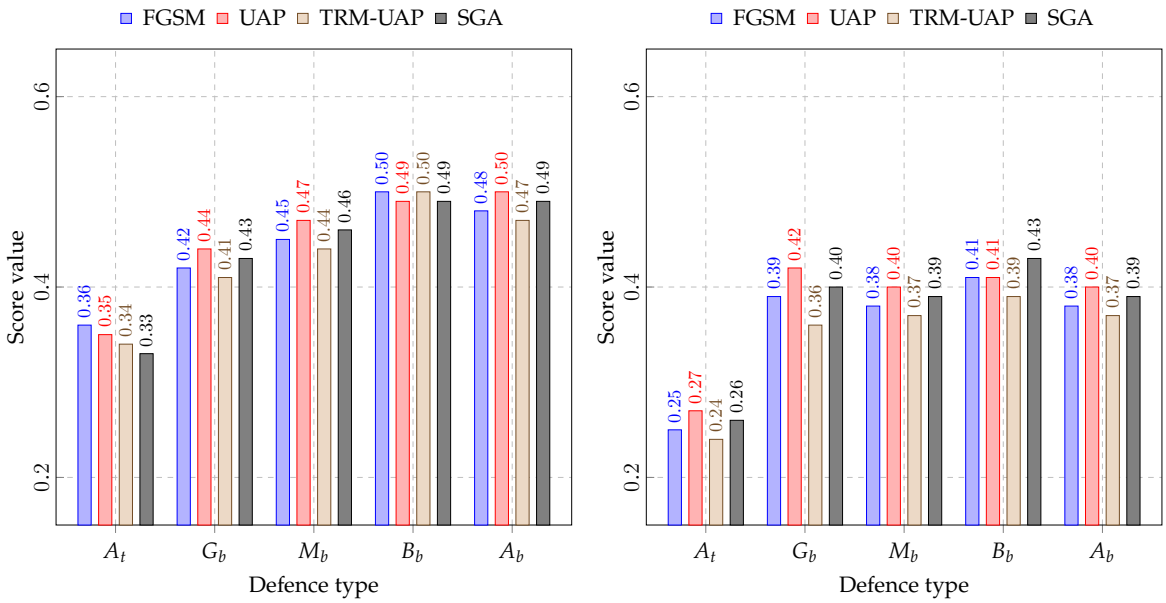
ial training (Ian J Goodfellow et al., 2015) and input transformation (Tian et al., 2024), the latter employing various types of blur. For a detailed explanation of how these defences were applied, refer to Section 4.3.3. With this context, Table 4.5 presents an overview of the FR values obtained for each defence, following the same format used in the previous evaluations.

Table 4.5: *MOSA-GAN robustness of different defence techniques based on model performance across various attack types — FGSM, UAP, TRM-UAP, and SGA — measured using the FR metric.*

Attack	AlexNet	ResNet152	ResNet18	VGG-16	VGG19
Adversarial Training					
FGSM	26.4±1.2	30.8±3.8	31.5±3.4	27.1±2.0	25.4±2.3
UAP	26.3±3.7	24.5±2.7	31.3±3.1	27.1±4.2	28.4±4.3
TRM-UAP	29.5±4.4	32.4±2.5	34.5±3.7	28.9±5.8	27.8±5.7
SGA	30.5±7.8	30.2±5.1	30.0±5.0	30.1±6.5	29.3±5.2
Input Transformation (*Gaussian Blur*)					
FGSM	65.6±1.4	70.7±4.3	71.4±3.4	63.4±2.4	61.9±2.5
UAP	61.9±4.0	58.1±2.8	71.8±3.1	63.5±4.4	66.3±4.6
TRM-UAP	67.1±4.7	73.0±2.7	74.1±3.8	65.6±6.2	64.5±6.2
SGA	68.9±8.7	68.5±5.7	68.3±5.4	68.4±6.9	66.4±5.8
Input Transformation (*Median Blur*)					
FGSM	63.3±1.4	68.3±4.5	68.9±3.3	61.2±2.3	59.8±2.5
UAP	59.8±4.0	56.1±2.9	69.4±3.2	61.3±4.3	64.0±4.6
TRM-UAP	64.9±4.6	70.5±2.6	71.6±3.7	63.3±6.1	62.3±6.2
SGA	66.6±8.6	66.2±5.6	65.9±5.3	66.0±6.8	64.2±5.7
Input Transformation (*Bilateral Blur*)					
FGSM	58.9±1.3	63.4±4.4	64.1±3.2	56.9±2.3	55.6±2.4
UAP	55.5±3.9	52.1±2.8	64.5±3.1	56.0±4.3	59.5±4.6
TRM-UAP	60.3±4.6	65.5±2.6	66.5±3.6	58.9±6.3	57.9±6.2
SGA	61.9±8.5	61.5±5.6	61.3±5.3	61.4±6.8	59.6±5.9
Input Transformation (*Affine Blur*)					
FGSM	60.3±1.4	65.0±4.5	65.7±3.3	58.3±2.4	57.0±2.5
UAP	56.9±4.0	53.5±2.8	66.1±3.2	58.4±4.4	61.0±4.7
TRM-UAP	61.8±4.7	67.2±2.7	68.2±3.6	60.3±6.2	59.4±6.3
SGA	63.4±8.6	63.1±5.6	62.9±5.4	63.0±6.9	61.2±5.8

Analysing the results presented in the table above, two distinct outcomes emerge across the two defence types. Adversarial training led to a substantial reduction in misclassification, with an average FR of 29.1%, representing a decrease of approximately 45.3% compared to the FR achieved by the proposed MOSA-GAN architecture. In contrast, input transformation defences produced only a minor reduction, as expected. Gaussian blur achieved an average FR of 66.9%, median blur 64.6%, bilateral blur 60.1%, and affine blur 61.6%. On average, these methods yielded an FR of 63.3%, corresponding to a decrease of only 11.1%, which is considerably lower than the reductions reported by Tian et al. (2024).

Similar to the previous evaluation, not only was the FR performance measured, but image quality and fidelity were also reassessed. Following the approach used in the previous study, the defences were evaluated against the FID and LPIPS metrics. The results, presented in Figure 4.4, are organised by defence method and averaged over the four attack types, perturbation magnitudes ξ, and the five target models.



(a) FID (b) LPIPS

Figure 4.4: *Correlation between average evaluation metrics (FID and LPIPS) and the defence type used for sample generation. Results are averaged across the tested models: AlexNet, VGG-16, VGG-19, ResNet18, and ResNet152. FID values were normalised to the [0,1] range by dividing by 100. Within the defence type, A_t denotes Adversarial Training, G_b denotes Gaussian Blur, M_b denotes Median Blur, B_b denotes Bilateral Blur, and A_b refers to Affine Blur.*

As the FR decreased in the results above, the FID and LPIPS metrics increased, particularly for the input transformation defence. In contrast, adversarial training maintained values comparable to those achieved by the proposed adversarial generative architecture, as expected, since its purpose is to enhance classifier robustness rather than alter the images. The increase in these metrics for the input transformation defence is justified, as the method modifies the images through blurring, which inevitably reduces the fidelity of the generated adversarial images.

4.4.4 Evaluating Adversarial Image Quality

After analysing the results obtained in the various stages of this evaluation study, a qualitative assessment was necessary to further validate the visual realism of the generated adversarial samples. This is particularly important because the goal is to produce generative samples that contain adversarial traits capable of misleading a DNN model while minimising perturbations in the image itself, rendering them imperceptible to humans. Figure 4.5 presents a comparison between real images and the generated images across the different discriminator weight configurations described in Section 4.4.2.

Figure 4.5: *Comparison of images generated using different discriminator weight configurations, including the proposed MOSA-GAN, against real images. The samples were generated with the SGA attack at a perturbation magnitude of $\xi = 0.10$, targeting the ResNet18 model.*

Regarding the generated images, the SuperstarGAN Adversarial approach introduces more visible perturbations, producing images that deviate noticeably from the original. In contrast, SuperstarGAN Vanilla closely resembles the source image, with only subtle distortions. Both SuperstarGAN Vanilla-Weighted and SuperstarGAN Adversarial-Weighted achieve a balance between noticeable changes and maintaining image integrity. The proposed MOSA-GAN architecture, however, achieves the best overall trade-off, effectively balancing image quality and fidelity in line with the evaluation metrics.

4.4.5 Discussion of Results

The results obtained in the previous section, as outlined earlier, can be grouped into four categories: the performance of the MOSA-GAN model in terms of misclassification, measured by the FR metric; the quality and fidelity of the generated images, assessed using the LPIPS and FID metrics alongside human-eye evaluation; the impact of varying discriminator weights; and finally, the robustness of the MOSA-GAN model against state-of-the-art defences. Each stage is discussed in detail below.

MOSA-GAN Performance

Regarding this first area of interest, and focusing exclusively on the FR metric, the MOSA-GAN model achieved results ranging from a maximum of 89% to a minimum of 66%. At first glance, and in comparison with the previous study (see Chapter 3), these may appear relatively weaker results. However, it is important to emphasise that the primary objective is not solely to maximise the FR, but rather to strike a balance between misclassification effectiveness and image quality.

Influence of Discriminator Weight Variation

In this test, we aimed to evaluate whether varying the weights of the two discriminators, $\mathcal{D}$ and $\mathcal{D}'$, affected the three metrics under study, namely FR, LPIPS, and FID. Six weight variations were tested, and the results confirmed the expected trend: increasing the weight of the adversarial discriminator $\mathcal{D}'$ led to higher FR, but at the cost of reduced image quality. Conversely, assigning all weight to the vanilla discriminator $\mathcal{D}$ resulted in much lower FR, which is suboptimal for the aims of this study. Therefore, the balance adopted in the proposed MOSA-GAN architecture — $\lambda_D = 0.65$ and $\lambda_{D'} = 0.35$ — represents, in our view, the best compromise to achieve the goals outlined in the previous area of discussion.

Image Quality and Fidelity

Another critical assessment in this study concerned image quality and fidelity. In the previous study, image quality was notably poor, making improvement in this area essential. By incorporating multi-objective optimisation and carefully controlling the discriminator weights, we achieved significantly better results. The FID metric reached a minimum (best) value of 24 and a maximum (worst) of 47. While this range does not indicate optimal quality, it does demonstrate good performance with only minor perturbations. For the LPIPS metric, values ranged from 0.19 to 0.40, which, although not perfect, are also strong results. Taken together with the FR outcomes, these findings suggest that the overall performance of MOSA-GAN is highly satisfactory.

Robustness of MOSA-GAN Against Defences

The final aspect of interest in this study concerns the performance of the model against state-of-the-art defences reported in the literature. Two distinct defence strategies were considered: adversarial training and input transformation. For the first, we observed a substantial reduction of approximately 45.3% in the FR. This outcome is expected,

since adversarial training aims to decrease the effectiveness of adversarial perturbations, thereby increasing the robustness of a given DNN when trained with the adversarial samples generated by MOSA-GAN. Conversely, the input transformation defence sought to mitigate adversarial perturbations by applying various blurring techniques prior to classification. While some reduction in the FR was observed, the average decrease was only 11.1%, considerably lower than that reported in the original work. This indicates that the adversarial images generated by MOSA-GAN are more resilient and demonstrate limited vulnerability to such defences.

4.5 Chapter Summary

The main objective of this study, building upon the conclusions of the previous work (see Chapter 3), was to design a novel generative architecture that employs multi-objective optimisation to enhance image quality and fidelity while sustaining a high FR in adversarial image generation. To this end, we proposed a new model, MOSA-GAN, which incorporates two discriminators with distinct objectives: one focused on producing high-fidelity images and the other on maximising adversarial traits.

Following the development of the MOSA-GAN architecture, a comprehensive evaluation methodology was established. This methodology extended that of the previous study by additionally assessing the influence of discriminator weight variation and the robustness of the model against state-of-the-art adversarial defences. Across the experimental evaluations, MOSA-GAN consistently achieved high FR values across all attacks and DNN models, with peak performance reaching approximately 89

In terms of image quality, the results demonstrated a clear improvement compared with the previous study, with favourable LPIPS and FID scores of around 0.23 and 25, respectively. The ablation studies on discriminator weights further confirmed that a balanced configuration between the vanilla and adversarial discriminators yields the best trade-off, producing high-quality images alongside strong adversarial effectiveness. Additionally, this work reaffirmed that perturbation magnitude directly influences all evaluation metrics: higher perturbations improve FR but degrade image quality, as reflected in both FID and LPIPS.

Finally, robustness testing against state-of-the-art defences showed that adversarial training was effective, reducing the FR of classification models by approximately 45.3%. This result highlights the potential of MOSA-GAN-generated images for training more robust classifiers. Conversely, input transformation defences proved less effective, achieving only an average reduction of 11.1% in FR, suggesting that adversarial samples produced by MOSA-GAN remain strong and resistant to such methods. Therefore, it can be concluded that the main objective of this study — maintaining high image quality and fidelity while achieving a high FR — was successfully accomplished.

5 Conclusion

It is widely acknowledged that AI will remain integral to society, and with it come frameworks and tools that assist daily activities. However, many such tools were developed without security by design; consequently, appropriate security measures are required. In the domain of CV, a prominent branch of AI, reliance on automated classification is common. Misclassification can have serious consequences in high-stakes applications such as finance or medicine, which motivates the work presented in this thesis.

This thesis aims to demonstrate empirically that it is possible to generate high-quality images that can mislead a given classification model and, in realistic use cases, cause adverse outcomes. Furthermore, the thesis investigates the constructive use of these images: they can be incorporated into defensive strategies to retrain classifiers and thereby improve their robustness against this class of attacks.

With this in mind, the work is divided into two parts. The first part tests the hypothesis that a pre-existing generative model can be used to produce adversarial images on demand; the second part presents a novel model, MOSA-GAN, designed for this purpose. In both studies, four attack types and five perturbation magnitudes were used, yielding twenty datasets employed to train the models. The generated images were evaluated against five pre-trained DNN classifiers.

In the first part, the pre-existing model achieved high misclassification rates, approximately 91% FR, but produced images of poor visual quality. Consequently, we developed the MOSA-GAN architecture, which leverages multi-objective optimisation and multiple discriminators to pursue two goals: high image quality and fidelity, and effective extraction of adversarial traits to increase misclassification.

Results for the MOSA-GAN were highly satisfactory. The model achieved a high FR of nearly 89% while maintaining good image quality, as measured by the FID and LPIPS metrics and validated through human-eye observations. To support these findings, additional tests were performed, including an ablation study on the impact of discriminator weights, an analysis of batch size effects on training losses across all components, an evaluation of different perturbation magnitudes on model performance, and, most importantly, an assessment of robustness against state-of-the-art defences in the literature.

From this last evaluation, it was concluded that the generated images can serve a dual purpose: not only to mislead DNN classifiers but also to strengthen them when used defensively. Specifically, by incorporating the generated adversarial images into the training process of classifiers, misclassification rates were reduced by up to 45.3%. Conversely, input transformation defences applied against the generated images achieved only a minor reduction in FR (11.1%), highlighting the strength and robustness of the adversarial examples produced by the MOSA-GAN model.

Based on the results obtained, it can be concluded that the objectives of this thesis were fully achieved: an existing generative model was evaluated for adversarial image generation, and a new model was designed and developed under rigorous and empirically grounded evaluation procedures. The proposed generative model, MOSA-GAN, demonstrates strong potential not only for malicious applications but, more importantly, as a defensive mechanism to enhance the robustness of classification systems, thereby addressing the dual-use nature of adversarial image generation. Furthermore, both sets of results were presented at international conferences. Specifically, the work in Chapter 3 was presented at a conference (Areia et al., 2025c), while the results from Chapter 4 were also presented at a conference (Areia et al., 2025a) and subsequently published (Areia et al., 2025b).

Nevertheless, certain limitations must be acknowledged. Hardware constraints restricted the training process, as larger GPU memory would have allowed for broader exploration of hyperparameter values. Similarly, employing the full dataset rather than a subset could have strengthened the evaluation, while expanding the decision boundaries of the attacks might have provided deeper insights. Further research could also involve testing a wider range of defences, experimenting with alternative generative architectures, and incorporating large language models for evaluation.

These limitations should not be regarded as shortcomings but rather as opportunities for future investigation. To support reproducibility and encourage further exploration, the entire codebase and underlying logic of this work have been released as open source.

In summary, all objectives established in this thesis were successfully achieved, contributing both to the scientific community through publications and presentations, and to the academic development of the author.

This page intentionally left blank.

Bibliography

Akhtar, Naveed et al. (2021). "Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey". In: *IEEE Access* 9, pp. 155161–155196. DOI: 10.1109/ACCESS.2021.3127960.

Akiba, Takuya et al. (2019). "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Albuquerque, Isabela et al. (2019). "Multi-objective training of generative adversarial networks with multiple discriminators". In: *International Conference on Machine Learning*, pp. 202–211.

Alec Radford Luke Metz, Soumith Chintala (Jan. 2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. DOI: 10.48550/arXiv.1511.06434. URL: http://arxiv.org/abs/1511.06434 (visited on 2025-02-11).

Areia, José, Leonel Santos, and Rogério Luís de C. Costa (2025a). *Balancing Image Quality and Attack Effectiveness in Multi-Objective Adversarial Image Generation*. URL: https://kdd2025.kdd.org/wp-content/uploads/2025/07/CameraReady-27.pdf.

Areia, José, Leonel Santos, and Rogério Luís de C. Costa (Sept. 2025b). *Balancing Image Quality and Attack Effectiveness in Multi-Objective Adversarial Image Generation*. https://ssrn.com/abstract=5457015. DOI: 10.2139/ssrn.5457015. URL: https://ssrn.com/abstract=5457015.

Areia, José, Leonel Santos, and Rogério Luís de C. Costa (2025c). "Fooling Rate and Perceptual Similarity: A Study on the Effectiveness and Quality of DCGAN-based Adversarial Attacks". In: *Availability, Reliability and Security*. Ed. by Mila Dalla Preda et al. Cham: Springer Nature Switzerland, pp. 420–430. ISBN: 978-3-032-00627-1.

Athalye, Anish, Nicholas Carlini, and David Wagner (July 2018a). *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*. DOI: 10.48550/arXiv.1802.00420. URL: http://arxiv.org/abs/1802.00420 (visited on 2024-07-15).

Athalye, Anish et al. (June 2018b). "Synthesizing Robust Adversarial Examples". In: *arXiv*. DOI: 10.48550/arXiv.1707.07397. URL: http://arxiv.org/abs/1707.07397 (visited on 2024-07-14).

Berahmand, Kamal et al. (2024). "Autoencoders and their applications in machine learning: a survey". In: *Artificial intelligence review* 57.2, p. 28.

Brendel, Wieland, Jonas Rauber, and Matthias Bethge (Feb. 2018). *Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models*. DOI: 10.48550/arXiv.1712.04248. URL: http://arxiv.org/abs/1712.04248 (visited on 2024-07-14).

Brundage, Miles et al. (2018). *The malicious use of artificial intelligence: Forecasting, prevention, and mitigation*. Tech. rep. arXiv preprint arXiv:1802.07228.

Brunner, Thomas et al. (Oct. 2019). "Guessing Smart: Biased Sampling for Efficient Black-Box Adversarial Attacks". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4957–4965. DOI: 10.1109/ICCV.2019.00506. URL: https://ieeexplore.ieee.org/document/9008375 (visited on 2024-07-15).

Burgess, Matt (Sept. 2024). "Apple Vision Pro's Eye Tracking Exposed What People Type". In: *WIRED*. Accessed: 2025-09-25. URL: https://www.wired.com/story/apple-vision-pro-persona-eye-tracking-spy-typing/.

Cai, Qi-Zhi, Chang Du, and Xiaolin Miao (2018). *Curriculum Adversarial Training*. URL: https://arxiv.org/abs/1805.04807.

Cao, Song and Ram Nevatia (Dec. 2016). "Exploring deep learning based solutions in fine grained activity recognition in the wild". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 384–389. DOI: 10.1109/ICPR.2016.7899664. URL: https://ieeexplore.ieee.org/document/7899664 (visited on 2024-07-13).

Carlini, Nicholas and David Wagner (Mar. 2017). *Towards Evaluating the Robustness of Neural Networks*. DOI: 10.48550/arXiv.1608.04644. URL: http://arxiv.org/abs/1608.04644 (visited on 2025-02-18).

Chakraborty, Anirban et al. (2021). "A survey on adversarial attacks and defences". en. In: *CAAI Transactions on Intelligence Technology* 6.1, pp. 25–45. ISSN: 2468-2322. DOI: 10.1049/cit2.12028. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12028 (visited on 2025-02-26).

Chen, Pin-Yu et al. (Nov. 2017). *ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models*. DOI: 10.1145/3128572.3140448. URL: http://arxiv.org/abs/1708.03999 (visited on 2024-07-14).

Chen, Shuangshuang and Wei Guo (2023). "Auto-encoders in deep learning—a review with new perspectives". In: *Mathematics* 11.8, p. 1777.

Chen, Xianjie and Alan Yuille (Nov. 2014). "Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations". In: *arXiv*. DOI: 10.48550/arXiv.1407.3399. URL: http://arxiv.org/abs/1407.3399 (visited on 2024-07-13).

Choi, Yunjey et al. (Sept. 2018a). *StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation*. DOI: 10.48550/arXiv.1711.09020. URL: http://arxiv.org/abs/1711.09020 (visited on 2025-02-23).

Choi, Yunjey et al. (2018b). *StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation*. eprint: 1711.09020. URL: https://arxiv.org/abs/1711.09020.

Cohen, Jeremy M., Elan Rosenfeld, and J. Zico Kolter (June 2019). *Certified Adversarial Robustness via Randomized Smoothing*. DOI: 10.48550/arXiv.1902.02918. URL: http://arxiv.org/abs/1902.02918 (visited on 2024-07-01).

Creswell, Antonia et al. (Jan. 2018). "Generative Adversarial Networks: An Overview". In: *IEEE Signal Processing Magazine* 35.1, pp. 53–65. ISSN: 1558-0792. DOI: 10.1109/msp.2017.2765202. URL: http://dx.doi.org/10.1109/MSP.2017.2765202.

Dalal, Navneet and Bill Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

Demir, Ugur and Gozde Unal (2018). *Patch-Based Image Inpainting with Generative Adversarial Networks*. eprint: 1803.07422. URL: https://arxiv.org/abs/1803.07422.

Diba, Ali et al. (Nov. 2016). "Weakly Supervised Cascaded Convolutional Networks". In: *arXiv*. DOI: 10.48550/arXiv.1611.08258. URL: http://arxiv.org/abs/1611.08258 (visited on 2024-07-13).

Ding, Rui et al. (2020). "BiGAN: collaborative filtering with bidirectional generative adversarial networks". In: *Proceedings of the 2020 SIAM international conference on data mining*. SIAM, pp. 82–90.

Dong, Yinpeng et al. (Mar. 2018). "Boosting Adversarial Attacks with Momentum". In: *arXiv*. DOI: 10.48550/arXiv.1710.06081. URL: http://arxiv.org/abs/1710.06081 (visited on 2024-05-21).

Doulamis, Nikolaos (Apr. 2018). "Adaptable deep learning structures for object labeling/tracking under dynamic visual environments". en. In: *Multimedia Tools and Applications* 77.8, pp. 9651–9689. ISSN: 1573-7721. DOI: 10.1007/s11042-017-5349-7. URL: https://doi.org/10.1007/s11042-017-5349-7 (visited on 2024-07-13).

Dumoulin, Vincent et al. (2017). *Adversarially Learned Inference*. eprint: 1606.00704. URL: https://arxiv.org/abs/1606.00704.

Durugkar, Ishan, Ian Gemp, and Sridhar Mahadevan (2017). *Generative Multi-Adversarial Networks*. eprint: 1611.01673. URL: https://arxiv.org/abs/1611.01673.

FastAI (2020). *Imagewoof*. URL: https://github.com/fastai/imagenette (visited on 2025-02-06).

Finlayson, Samuel G et al. (2019). "Adversarial attacks on medical machine learning". In: *Science* 363.6433, pp. 1287–1289.

Gemini Robotics Team et al. (2025). *Gemini Robotics: Bringing AI into the Physical World*. eprint: 2503.20020. URL: https://arxiv.org/abs/2503.20020.

Gong, Zhitao, Wenlu Wang, and Wei-Shinn Ku (2017). *Adversarial and Clean Data Are Not Twins*. International Conference on Security and Privacy in Communication Systems (SecureComm). URL: https://arxiv.org/abs/1707.04319.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2015). "Explaining and harnessing adversarial examples". In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. arXiv: 1406.2661. URL: https://arxiv.org/abs/1406.2661.

Gu, Shixiang and Luca Rigazio (2015). *Towards Deep Neural Network Architectures Robust to Adversarial Examples*. ICLR Workshop. URL: https://arxiv.org/abs/1412.5068.

He, Cheng et al. (2020). "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)". In: *IEEE transactions on cybernetics* 51.6, pp. 3129–3142.

He, Kaiming et al. (2016a). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

He, Kaiming et al. (2016b). *Deep Residual Learning for Image Recognition*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). URL: https://doi.org/10.1109/CVPR.2016.90.

Heusel, Martin et al. (2017). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html (visited on 2025-02-20).

Hinton, Geoffrey E, Sara Sabour, and Nicholas Frosst (2018). *Matrix Capsules with EM Routing*. International Conference on Learning Representations (ICLR). URL: https://arxiv.org/abs/1710.09829.

Huang, Ling et al. (Oct. 2011). "Adversarial machine learning". In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. AISec '11. New York, NY, USA: Association for Computing Machinery, pp. 43–58. ISBN: 9781450310031. DOI: 10.1145/2046684.2046692. URL: https://dl.acm.org/doi/10.1145/2046684.2046692 (visited on 2023-09-28).

Hubel, D. H. and T. N. Wiesel (Jan. 1962). "Receptive Fields, Binocular Interaction, and Functional Architecture in the Cat's Visual Cortex". In: *The Journal of Physiology* 160.1, pp. 106–154. DOI: 10.1113/jphysiol.1962.sp006837.

Jakubovitz, Daniel and Raja Giryes (2018). *Improving DNN Robustness to Adversarial Attacks using Jacobian Regularization*. URL: https://arxiv.org/abs/1803.08680.

Jenkins, John and Kaushik Roy (May 2024). "Exploring deep convolutional generative adversarial networks (DCGAN) in biometric systems: a survey study". en. In: *Discover Artificial Intelligence* 4.1, p. 42. ISSN: 2731-0809. DOI: 10.1007/s44163-024-00138-z. URL: https://doi.org/10.1007/s44163-024-00138-z (visited on 2025-02-23).

Kingma, Diederik P, Max Welling, et al. (2019). "An introduction to variational autoencoders". In: *Foundations and Trends in Machine Learning* 12.4, pp. 307–392.

Kingma, Diederik P. and Jimmy Ba (Jan. 2017). *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/arXiv.1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 2025-05-13).

Ko, Kanghyeok, Taesun Yeom, and Minhyeok Lee (2023). "SuperstarGAN: Generative adversarial networks for image-to-image translation in large-scale domains". In: *Neural Networks* 162, pp. 330–339. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2023.02.042. URL: https://www.sciencedirect.com/science/article/pii/S0893608023001144.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS 2012)*. Vol. 25. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Krumm, John (2007). "Inference Attacks on Location Tracks". en. In: *Pervasive Computing*. Ed. by Anthony LaMarca, Marc Langheinrich, and Khai N. Truong. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 127–143. ISBN: 9783540720379. DOI: 10.1007/978-3-540-72037-9_8.

Kurakin, Alexey, Ian Goodfellow, and Samy Bengio (Feb. 2017). *Adversarial Machine Learning at Scale*. DOI: 10.48550/arXiv.1611.01236. URL: http://arxiv.org/abs/1611.01236 (visited on 2024-07-14).

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). *Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles*. Advances in Neural Information Processing Systems (NeurIPS). URL: https://arxiv.org/abs/1612.01474.

Lee, Minhyeok and Junhee Seok (2019a). "Controllable Generative Adversarial Network". In: *IEEE Access* 7, pp. 28158–28169. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2899108. URL: https://ieeexplore.ieee.org/document/8641270 (visited on 2025-02-23).

Lee, Minhyeok and Junhee Seok (2019b). "Controllable Generative Adversarial Network". In: *IEEE Access* 7, pp. 28158–28169. DOI: 10.1109/ACCESS.2019.2899108.

Li, Jianbin et al. (Apr. 2025). "DPP-GAN: A decentralized and privacy-preserving GAN system for collaborative smart meter data generation". In: *Energy and Buildings* 333, p. 115489.

ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2025.115489. URL: https://www.sciencedirect.com/science/article/pii/S0378778825002191 (visited on 2025-02-23).

Lin, Liang et al. (June 2016). "A Deep Structured Model with Radius–Margin Bound for 3D Human Activity Recognition". en. In: *International Journal of Computer Vision* 118.2, pp. 256–273. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0876-z. URL: https://doi.org/10.1007/s11263-015-0876-z (visited on 2024-07-13).

Liu, Bingqi et al. (2022). "Application of an Improved DCGAN for Image Generation". en. In: *Mobile Information Systems* 2022.1, p. 9005552. ISSN: 1875-905X. DOI: 10.1155/2022/9005552. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/9005552 (visited on 2025-02-23).

Liu, Bo et al. (Mar. 2021). "When Machine Learning Meets Privacy: A Survey and Outlook". In: *ACM Computing Surveys* 54.2, 31:1–31:36. ISSN: 0360-0300. DOI: 10.1145/3436755. URL: https://doi.org/10.1145/3436755 (visited on 2023-10-06).

Liu, Ximeng et al. (2021). "Privacy and Security Issues in Deep Learning: A Survey". In: *IEEE Access* 9, pp. 4566–4593. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3045078. URL: https://ieeexplore.ieee.org/abstract/document/9294026 (visited on 2023-10-04).

Liu, Xuannan et al. (Aug. 2023). *Enhancing Generalization of Universal Adversarial Perturbation through Gradient Aggregation*. DOI: 10.48550/arXiv.2308.06015. URL: http://arxiv.org/abs/2308.06015 (visited on 2024-05-14).

Liu, Yiran et al. (Oct. 2023). "TRM-UAP: Enhancing the Transferability of Data-Free Universal Adversarial Perturbation via Truncated Ratio Maximization". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. ISSN: 2380-7504, pp. 4739–4748. DOI: 10.1109/ICCV51070.2023.00439. URL: https://ieeexplore.ieee.org/document/10377636 (visited on 2024-06-29).

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (Mar. 2015). "Fully Convolutional Networks for Semantic Segmentation". In: *arXiv*. DOI: 10.48550/arXiv.1411.4038. URL: http://arxiv.org/abs/1411.4038 (visited on 2024-07-13).

Lowe, David G. (Nov. 2004). "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2, pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

Lu, Yang (2019). "Artificial intelligence: a survey on evolution, models, applications and future trends". In: *Journal of management analytics* 6.1, pp. 1–29.

Madry, Aleksander et al. (2018). "Towards deep learning models resistant to adversarial attacks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Meng, Dongyu and Hao Chen (2017). *MagNet: A Two-Pronged Defense against Adversarial Examples*. ACM SIGSAC Conference on Computer and Communications Security (CCS). URL: https://arxiv.org/abs/1705.09064.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard (July 2016). *DeepFool: a simple and accurate method to fool deep neural networks*. DOI: 10.48550/arXiv.1511.04599. URL: http://arxiv.org/abs/1511.04599 (visited on 2024-05-14).

Moosavi-Dezfooli, Seyed-Mohsen et al. (Mar. 2017). *Universal adversarial perturbations*. DOI: 10.48550/arXiv.1610.08401. URL: http://arxiv.org/abs/1610.08401 (visited on 2024-05-14).

Mopuri, Konda Reddy, Aditya Ganeshan, and R. Venkatesh Babu (July 2018). *Generalizable Data-free Objective for Crafting Universal Adversarial Perturbations*. DOI: 10.48550/arXiv.1801.08092. URL: http://arxiv.org/abs/1801.08092 (visited on 2024-07-15).

Mopuri, Konda Reddy, Utsav Garg, and R. Venkatesh Babu (July 2017). *Fast Feature Fool: A data independent approach to universal adversarial perturbations*. DOI: 10.48550/arXiv.1707.05572. URL: http://arxiv.org/abs/1707.05572 (visited on 2024-07-15).

Nie, Weili et al. (2022). *Diffusion Models for Adversarial Purification*. URL: https://arxiv.org/abs/2205.07460.

Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han (May 2015). "Learning Deconvolution Network for Semantic Segmentation". In: *arXiv*. DOI: 10.48550/arXiv.1505.04366. URL: http://arxiv.org/abs/1505.04366 (visited on 2024-07-13).

Odena, Augustus, Christopher Olah, and Jonathon Shlens (July 2017). "Conditional Image Synthesis with Auxiliary Classifier GANs". en. In: *Proceedings of Machine Learning Research*. PMLR, pp. 2642–2651. URL: https://proceedings.mlr.press/v70/odena17a.html (visited on 2025-02-23).

Ouyang, Wanli et al. (July 2017). "DeepID-Net: Object Detection with Deformable Part Based Convolutional Neural Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.7, pp. 1320–1334. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2016.2587642. URL: https://ieeexplore.ieee.org/document/7506134 (visited on 2024-07-13).

Paniagua, Thomas, Chinmay Savadikar, and Tianfu Wu (2025). "Adversarial Perturbations Are Formed by Iteratively Learning Linear Combinations of the Right Singular Vectors of the Adversarial Jacobian". In: *Forty-second International Conference on Machine Learning*. URL: https://openreview.net/forum?id=3q6T4hQ4Dy.

Pannu, Avneet (2018). "Artificial intelligence and its application in different areas". In: *Artificial Intelligence* 4.10, pp. 79–84.

Papernot, Nicolas et al. (2016). "The limitations of deep learning in adversarial settings". In: *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, pp. 372–387.

Parekh, Darsh et al. (2022). "A review on autonomous vehicles: Progress, methods and challenges". In: *Electronics* 11.14, p. 2162.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *arXiv preprint arXiv:1912.01703*.

Poursaeed, Omid et al. (2018). *Generative Adversarial Perturbations*. eprint: 1712.02328. URL: https://arxiv.org/abs/1712.02328.

Racine, Eric, Wren Boehlen, and Matthew Sample (2019). "Healthcare uses of artificial intelligence: Challenges and opportunities for growth". In: *Healthcare management forum*. Vol. 32. 5. SAGE Publications Sage CA, pp. 272–275.

Al-Rubaie, Mohammad and J. Morris Chang (Mar. 2019). "Privacy-Preserving Machine Learning: Threats and Solutions". In: *IEEE Security & Privacy* 17.2, pp. 49–58. ISSN: 1558-4046. DOI: 10.1109/MSEC.2018.2888775. URL: https://ieeexplore.ieee.org/abstract/document/8677282 (visited on 2023-10-04).

Russakovsky, Olga et al. (Jan. 2015). *ImageNet Large Scale Visual Recognition Challenge*. DOI: 10.48550/arXiv.1409.0575. URL: http://arxiv.org/abs/1409.0575 (visited on 2024-07-15).

Samangouei, Pouya, Mohammad Kabkab, and Rama Chellappa (2018). *Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models*. International Conference on Learning Representations (ICLR). URL: https://arxiv.org/abs/1805.06507.

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICLR 2015)*. URL: https://arxiv.org/abs/1409.1556.

Song, Yang et al. (2018). *PixelDefend: Leveraging Generative Models to Understand and Defend Against Adversarial Examples*. International Conference on Learning Representations (ICLR). URL: https://arxiv.org/abs/1805.06605.

Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai (Oct. 2019). "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Transactions on Evolutionary Computation* 23.5, pp. 828–841. ISSN: 1941-0026. DOI: 10.1109/TEVC.2019.2890858. URL: https://ieeexplore.ieee.org/document/8601309 (visited on 2024-07-15).

Sun, Hui, Siman Wu, and Lijun Ma (Aug. 2024). "Adversarial attacks on GAN-based image fusion". In: *Information Fusion* 108, p. 102389. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2024.102389. URL: https://www.sciencedirect.com/science/article/pii/S1566253524001672 (visited on 2025-05-14).

Szegedy, Christian et al. (2014). "Intriguing properties of neural networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Szegedy, Christian et al. (Dec. 2015). *Rethinking the Inception Architecture for Computer Vision*. DOI: 10.48550/arXiv.1512.00567. URL: http://arxiv.org/abs/1512.00567 (visited on 2025-03-07).

Szegedy, Christian et al. (2016). *Rethinking the Inception Architecture for Computer Vision*. URL: https://arxiv.org/abs/1512.00567.

Szeliski, Richard (2022). *Computer vision: algorithms and applications*. Springer Nature.

Tan, Mingxing and Quoc V. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*. Vol. 97, pp. 6105–6114. URL: http://proceedings.mlr.press/v97/tan19a.html.

Tian, Pu et al. (2024). "Evaluating Impact of Image Transformations on Adversarial Examples". In: *IEEE Access* 12, pp. 186217–186228. DOI: 10.1109/ACCESS.2024.3487479.

Tiwari, Sumit (2016). "An introduction to QR code technology". In: *2016 international conference on information technology (ICIT)*. IEEE, pp. 39–44.

Toshev, Alexander and Christian Szegedy (June 2014). "DeepPose: Human Pose Estimation via Deep Neural Networks". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660. DOI: 10.1109/CVPR.2014.214. URL: http://arxiv.org/abs/1312.4659 (visited on 2024-07-13).

Tu, Chun-Chen et al. (Jan. 2020). *AutoZOOM: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-box Neural Networks*. DOI: 10.48550/arXiv.1805.11770. URL: http://arxiv.org/abs/1805.11770 (visited on 2024-07-15).

Van Veldhuizen, David A, Gary B Lamont, et al. (1998). "Evolutionary computation and convergence to a pareto front". In: *Late breaking papers at the genetic programming 1998 conference*, pp. 221–228.

Viola, Jairo, YangQuan Chen, and Jing Wang (Jan. 2021). "FaultFace: Deep Convolutional Generative Adversarial Network (DCGAN) based Ball-Bearing failure detection method". In: *Information Sciences* 542, pp. 195–211. ISSN: 0020-0255. DOI: 10.1016/j.ins.2020.06.060. URL: https://www.sciencedirect.com/science/article/pii/S0020025520306484 (visited on 2025-02-23).

Voulodimos, Athanasios et al. (2018). "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018.1, p. 7068349.

Wang, Zhenyang et al. (2019). "Design and implementation of vehicle unlocking system based on face recognition". In: *2019 34rd Youth academic annual conference of chinese association of automation (YAC)*. IEEE, pp. 121–126.

Xiao, Chaowei et al. (Feb. 2019). *Generating Adversarial Examples with Adversarial Networks*. DOI: 10.48550/arXiv.1801.02610. URL: http://arxiv.org/abs/1801.02610.

Xu, Hongyang et al. (2020). *Robustness of Deep Learning Models Against Adversarial Attacks: A Survey*. arXiv preprint arXiv:2007.10707. URL: https://arxiv.org/abs/2007.10707.

Xu, Weilin, David Evans, and Yanjun Qi (2018). *Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks*. Network and Distributed System Security Symposium (NDSS). URL: https://arxiv.org/abs/1704.01155.

Zhang, Caiming and Yang Lu (2021). "Study on artificial intelligence: The state of the art and future prospects". In: *Journal of Industrial Information Integration* 23, p. 100224.

Zhang, Chiyu et al. (Sept. 2025). "Adversarial Attacks of Vision Tasks in the Past 10 Years: A Survey". In: *ACM Comput. Surv.* 58.2. ISSN: 0360-0300. DOI: 10.1145/3743126. URL: https://doi.org/10.1145/3743126.

Zhang, Richard et al. (Apr. 2018). *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. DOI: 10.48550/arXiv.1801.03924. URL: http://arxiv.org/abs/1801.03924 (visited on 2025-02-20).

Zhang, Xiaoyuan et al. (2023). "Hypervolume maximization: A geometric view of pareto set learning". In: *Advances in Neural Information Processing Systems* 36, pp. 38902–38929.

Zhao, Xia et al. (2024). "A Review of Convolutional Neural Networks in Computer Vision". In: *Artificial Intelligence Review* 57.4, p. 99. URL: https://doi.org/10.1007/s10462-024-10721-6.

Zhou, Tinghui et al. (2016). *Learning Dense Correspondence via 3D-guided Cycle Consistency*. eprint: 1604.05383. URL: https://arxiv.org/abs/1604.05383.

Zhu, Jun-Yan et al. (2020). *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. eprint: 1703.10593. URL: https://arxiv.org/abs/1703.10593.

This page intentionally left blank.

Appendices

This page intentionally left blank.

A

MOSA-GAN Generative Model Code Implementations

This appendix chapter is included solely to present partial code implementations of the proposed MOSA-GAN model outlined in Chapter 4. Please refer to the caption of each listing to identify the corresponding class or function being presented.

Listing 1: *Implementation of the Vanilla Discriminator class.*

```
class Discriminator(nn.Module):
    def __init__(self, image_size=128, conv_dim=64, num_classes=5, repeat_num=6):
        super(Discriminator, self).__init__()
        layers = []
        layers.append(spectral_norm(nn.Conv2d(3, conv_dim, kernel_size=4, stride=2,
        ↪  padding=1)))
        layers.append(nn.LeakyReLU(0.01))

        curr_dim = conv_dim
        for i in range(1, repeat_num):
            layers.append(spectral_norm(nn.Conv2d(curr_dim, curr_dim*2, kernel_size=4,
            ↪  stride=2, padding=1)))
            layers.append(nn.LeakyReLU(0.01))
            curr_dim = curr_dim * 2

        kernel_size = int(image_size / np.power(2, repeat_num))
        self.main = nn.Sequential(*layers)
        self.conv1 = spectral_norm(nn.Conv2d(curr_dim, 1, kernel_size=3, stride=1,
        ↪  padding=1, bias=False))

    def forward(self, x):
        h = self.main(x)
        out_src = self.conv1(h)
        return out_src
```

Listing 2: *Implementation of the Adversarial Discriminator class.*

```
class AdversarialDiscriminator(nn.Module):
    def __init__(self, image_size=128, conv_dim=64, num_classes=5, repeat_num=6):
        super(AdversarialDiscriminator, self).__init__()
        layers = []
        layers.append(spectral_norm(nn.Conv2d(3, conv_dim, kernel_size=4, stride=2,
        ↪  padding=1)))
        layers.append(nn.LeakyReLU(0.01))

        curr_dim = conv_dim
        for i in range(1, repeat_num):
            layers.append(spectral_norm(nn.Conv2d(curr_dim, curr_dim * 2, kernel_size=4,
            ↪  stride=2, padding=1)))
            layers.append(nn.LeakyReLU(0.01))
            curr_dim *= 2

        kernel_size = int(image_size / np.power(2, repeat_num))
        self.main = nn.Sequential(*layers)
        self.fc = spectral_norm(nn.Linear(curr_dim * kernel_size * kernel_size,
        ↪  num_classes))

    def forward(self, x):
        h = self.main(x)
        h = h.view(h.size(0), -1)
        return self.fc(h)
```

Listing 3: *Implementation of the Generator class.*

```
class Generator(nn.Module):
    def __init__(self, conv_dim=64, c_dim=5, repeat_num=6):
        super(Generator, self).__init__()
        layers = []
        layers.append(nn.Conv2d(3+c_dim, conv_dim, kernel_size=7, stride=1, padding=3,
        ↪  bias=False))
        layers.append(nn.InstanceNorm2d(conv_dim, affine=True, track_running_stats=True))
        layers.append(nn.ReLU(inplace=True))

        curr_dim = conv_dim
        for i in range(2):
            layers.append(nn.Conv2d(curr_dim, curr_dim*2, kernel_size=4, stride=2,
            ↪  padding=1, bias=False))
            layers.append(nn.InstanceNorm2d(curr_dim*2, affine=True,
            ↪  track_running_stats=True))
            layers.append(nn.ReLU(inplace=True))
            curr_dim = curr_dim * 2

        for i in range(repeat_num):
            layers.append(ResidualBlock(dim_in=curr_dim, dim_out=curr_dim))

```

```
        for i in range(2):
            layers.append(nn.ConvTranspose2d(curr_dim, curr_dim//2, kernel_size=4,
            ↪  stride=2, padding=1, bias=False))
            layers.append(nn.InstanceNorm2d(curr_dim//2, affine=True,
            ↪  track_running_stats=True))
            layers.append(nn.ReLU(inplace=True))
            curr_dim = curr_dim // 2

        layers.append(nn.Conv2d(curr_dim, 3, kernel_size=7, stride=1, padding=3,
        ↪  bias=False))
        layers.append(nn.Tanh())
        self.main = nn.Sequential(*layers)

    def forward(self, x, c):
        c = c.view(c.size(0), c.size(1), 1, 1)
        c = c.repeat(1, 1, x.size(2), x.size(3))
        x = torch.cat([x, c], dim=1)
        return self.main(x)
```

Listing 4: *Implementation of the Classifier class.*

```
class Classifier(nn.Module):
    def __init__(self, image_size=128, conv_dim=64, c_dim=40, repeat_num=6):
        super(Classifier, self).__init__()
        layers = []
        layers.append(nn.Conv2d(3, conv_dim, kernel_size=4, stride=2, padding=1))
        layers.append(nn.LeakyReLU(0.01))

        curr_dim = conv_dim
        for i in range(1, repeat_num):
            layers.append(nn.Conv2d(curr_dim, curr_dim*2, kernel_size=4, stride=2,
            ↪  padding=1))
            layers.append(nn.LeakyReLU(0.01))
            curr_dim = curr_dim * 2

        kernel_size = int(image_size / np.power(2, repeat_num))
        self.main = nn.Sequential(*layers)
        self.conv2 = nn.Conv2d(curr_dim, c_dim, kernel_size=kernel_size, bias=False)

    def forward(self, x):
        h = self.main(x)
        out_cls = self.conv2(h)
        return out_cls.view(out_cls.size(0), out_cls.size(1))
```

Listing 5: *Implementation of the Encoder class.*

```
class Encoder(nn.Module):
    def __init__(self, image_size=128, conv_dim=64, c_dim=5, repeat_num=6):
```

```
        super(Encoder, self).__init__()
        layers = []
        layers.append(nn.Conv2d(3, conv_dim, kernel_size=4, stride=2, padding=1))
        layers.append(nn.ReLU())

        curr_dim = conv_dim
        for i in range(1, repeat_num):
            layers.append(nn.Conv2d(curr_dim, curr_dim*2, kernel_size=4, stride=2,
            ↪  padding=1))
            layers.append(nn.ReLU())
            curr_dim = curr_dim * 2

        kernel_size = int(image_size / np.power(2, repeat_num))
        self.main = nn.Sequential(*layers)
        self.out = nn.Conv2d(curr_dim, c_dim, kernel_size=kernel_size, bias=False)

    def forward(self, x):
        h = self.main(x)
        c_pred = self.out(h)
        return c_pred.view(c_pred.size(0), -1)
```

B

Adversarial DCGAN Generative Model Code Implementations

This appendix chapter is included solely to present partial code implementations of the adversarial DCGAN model outlined in Chapter 3. Please refer to the caption of each listing to identify the corresponding class or function being presented.

Listing 6: *Implementation of the Discriminator class.*

```
class Discriminator(nn.Module):
    def __init__(self, gpu_number, feature_map_d, channel_number, latent_vector_z):
        super(Discriminator, self).__init__()
        self.gpu_number = gpu_number
        self.feature_map_d = feature_map_d
        self.channel_number = channel_number
        self.latent_vector_z = latent_vector_z
        self.main = nn.Sequential(
            nn.Conv2d(channel_number, feature_map_d, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_d, feature_map_d*2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_d*2),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_d*2, feature_map_d*4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_d*4),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_d*4, feature_map_d*8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_d*8),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_d*8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )
```

```
27
28      def forward(self, input):
29          return self.main(input)
```

Listing 7: *Implementation of the Generator class.*

```
1   class Generator(nn.Module):
2       def __init__(self, gpu_number, feature_map_g, channel_number, latent_vector_z):
3           super(Generator, self).__init__()
4           self.gpu_number = gpu_number
5           self.feature_map_g = feature_map_g
6           self.channel_number = channel_number
7           self.latent_vector_z = latent_vector_z
8           self.main = nn.Sequential(
9               nn.ConvTranspose2d(latent_vector_z, feature_map_g*8, 4, 1, 0, bias=False),
10              nn.BatchNorm2d(feature_map_g*8),
11              nn.ReLU(True),
12
13              nn.ConvTranspose2d(feature_map_g*8, feature_map_g * 4, 4, 2, 1, bias=False),
14              nn.BatchNorm2d(feature_map_g*4),
15              nn.ReLU(True),
16
17              nn.ConvTranspose2d(feature_map_g*4, feature_map_g*2, 4, 2, 1, bias=False),
18              nn.BatchNorm2d(feature_map_g*2),
19              nn.ReLU(True),
20
21              nn.ConvTranspose2d(feature_map_g*2, feature_map_g, 4, 2, 1, bias=False),
22              nn.BatchNorm2d(feature_map_g),
23              nn.ReLU(True),
24
25              nn.ConvTranspose2d(feature_map_g, channel_number, 4, 2, 1, bias=False),
26              nn.Tanh()
27          )
28
29      def forward(self, input):
30          return self.main(input)
```

Listing 8: *Implementation of the Encoder class.*

```
1   class Encoder(nn.Module):
2       def __init__(self, feature_map_e, channel_number, latent_vector_z):
3           super(Encoder, self).__init__()
4           self.feature_map_e = feature_map_e
5           self.channel_number = channel_number
6           self.latent_vector_z = latent_vector_z
7           self.main = nn.Sequential(
8               nn.Conv2d(channel_number, feature_map_e, 4, 2, 1, bias=False),
9               nn.BatchNorm2d(feature_map_e),
```

```
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_e, feature_map_e*2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_e*2),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_e*2, feature_map_e*4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_e*4),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_e*4, feature_map_e*8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_e*8),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(feature_map_e*8, latent_vector_z, 4, 1, 0, bias=False),
        )

    def forward(self, x):
        return self.main(x).view(x.size(0), -1)
```

This page intentionally left blank.