Chapter 1

Introduction

In recent years, Generative Adversarial Networks (GANs) have emerged as a powerful tool for generating realistic data. GANs were introduced as a type of generative model based on game theory in which two networks compete in a min-max game to mimic a data distribution using Artificial Neural Networks (ANNs). In a two-player game, each player aims to maximize their own objective function, which often leads to a situation where the objectives of the players are in direct conflict with each other.

GANs have had a significant impact on the field of artificial intelligence by enabling the creation of large amounts of realistic data. This breakthrough has sparked a surge of research in various areas such as language and image generation, image-to-image translation, image generation from textual description, video generation, and more. GANs have achieved remarkable results in these domains, surpassing previous benchmarks and setting new standards for artificial intelligence-generated data.

However, GANs suffer from instability during training, leading to issues such as mode collapse. To solve these problems, recently a variety of GAN architectures has been developed that are adapts to specific applications. This survey aims to provide an overview of the latest GAN architectures and their applications, as well as how they address the challenges of GAN training. The study will also summarize commonly used metrics for measuring GAN performance and propose a classification of GANs based on their applications.



Figure 1.1: Basic Architecture of GANs.

Chapter 2

Basic Concepts/Literature Review

2.1 – Generative Adversarial Networks

Generative Adversarial Networks (GANs) consists of two main components:

* A discriminator D is responsible for evaluating a given sample to determine whether it is real or fake. It essentially acts as a critic and tries to distinguish between real samples from the training dataset and synthetic samples generated by the GAN's generator.
* A generator G is responsible for creating synthetic samples that closely resemble the real samples from the training dataset. The generator takes a noise variable input, denoted as z, and uses it to generate synthetic samples.

During the process of training, these two models compete to maximize their own objective function forming a zero-sum game. The generator G tries to deceive the discriminator, while the discriminator D is optimized to accurately classify each sample as original or not. The loss function of this game is formulated as a minimization of the Generator and maximization of the Discriminator's loss.

We want to maximize of the discriminator D’s classifications over real data. Meanwhile, minimizing the probability given a fake sample , by maximizing .

The generator is trained to minimize increasing the chances of D producing a higher number a fake data. When combining both aspects together, D and G are playing a minimax game in which we should optimize the following loss function:



( does not impact the gradient descent updates.)

* : Distribution of the noise input
* : Generator’s distribution of the data
* : Distribution of the real sample

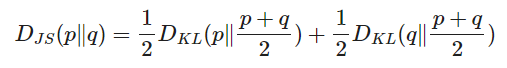
2.2 – Training Problems

Despite the impressive results that Generative Adversarial Networks (GANs) have achieved in generating realistic images, the training process can be challenging. GANs are notorious for being slow and unstable during the training phase.

* **Hard to achieve Nash Equilibrium:** The issue with GAN's training process based on gradient descent is that it involves two models learning simultaneously to find a Nash equilibrium in a two-player game. However, each model updates its cost function independently without taking into account the other player's behavior. Updating the gradients of both models concurrently does not guarantee convergence to a stable equilibrium.
* **Mode Collapse:** GANs can sometimes fail to generate diverse and realistic data during training, which is referred to as mode collapse. This means that the generator may produce the same outputs for different inputs, resulting in a limited variety of synthesized data. This problem can occur in a complete form, where all outputs are identical, or in a partial form, where a large number of outputs share similar characteristics.
* **Vanishing Gradient:** This can occur during GAN training when the discriminator becomes too accurate i.e or too inaccurate i.e . This makes the loss function decline to zero leading to no gradient, hence updating the loss is not possible. If the discriminator is too accurate, it can easily differentiate between real and synthesized data, resulting in a loss function that approaches zero and providing little feedback to the generator. This leads to gradients that are close to zero, which slows down the learning process or even causes it to jam. If the discriminator is too inaccurate, it will not be able to differentiate between the two types of samples effectively, also provides the generator with useless information and causes the loss function to fall to zero. In either case, the GAN training faces a dilemma where it becomes very tough to achieve balanced learning between the generator and discriminator.
* **Instability and Stopping Problem:** The objective of both the generator and discriminator networks is opposite, and even small changes in one network can lead to significant changes in the other. This can instability during training, which will generate further instability, making the training period longer. Many GAN architectures focus on stabilizing their training to improve their performance. Furthermore, GAN training are difficult to know if the models are optimized as they do not follow a monotonically decreasing loss function.

2.3 – Jensen-Shannon Divergence & Wasserstein Distance

Jensen-Shannon Divergence (JS divergence) is a symmetric and smoother alternative to the Kullback-Leibler Divergence. It measures the correlation between two distributions in a fuzzy way. Its use in GANs has been successful, as it replaced the asymmetric KL divergence used in the traditional maximum-likelihood approach.



Wasserstein Distance, (EM distance), is a mathematical metric used to measure the distance between two probability distributions. It takes into account the underlying geometry and structure of the distributions being compared. It calculates the minimum amount of work required to transform one distribution into another, where "work" is defined as the amount of probability mass that needs to be moved from one point to another.

2.4 – Evaluation Metrics

Evaluation metrics are used to measure the quality and performance of GAN models. However, due to the unique nature of GANs, there is no universal metric that can be used to evaluate all GAN applications. Instead, researchers have developed different metrics, each with its own strengths and weaknesses, to measure different aspects of GAN performance. The most widely used evaluation metrics include the Inception Score (IS) and its variants, Multi-scale Structural Similarity for Image Quality (MS-SSIM), and Classifier Two-sample Test (C2ST).

The IS measures uses a pre-trained neural network classifier (Inception v3) to calculate the probabilities of the generated samples. It assumes that low entropy and high-quality data are correlated, but it cannot handle mode collapse, where all generated samples are practically the same.

Another metric commonly used with IS or its variations is the Multi-Scale Structural Similarity Index (MS-SSIM), which compares the similarity between the real and synthesized datasets based on the geometry and structure of the image.

The Classifier Two-Sample Test (C2ST) is a method for comparing two datasets using a binary classifier to determine whether they come from the same distribution. C2ST can use a variety of classifiers, including 1-Nearest Neighbor and Neural Networks. It is useful in detecting statistical differences in high-dimensional datasets without making assumptions about their underlying distributions.

Overall, researchers use a combination of these metrics to evaluate GAN performance and to have a wider view of the model's strengths and weaknesses.

2.5 – Improved Approaches to GAN Training

The following suggestions are recommended to improve the training of Generative Adversarial Networks (GANs). A paper titled "Improve Techniques for Training GANs" that proposes five techniques to speed up the convergence of GAN training. The last two methods are proposed in another paper titled "Towards principled methods for training generative adversarial networks" and address the problem of disjoint distributions.

* **Feature Matching**, involves optimizes the D to compare the G's output with the expected results of real data. This is done by defining a new loss function that calculates the difference between the two sets of statistics.
* **Minibatch Discrimination**, ivolves dividing a minibatch of samples into subsets and computing the distance between each subset using a learned metric. The distance measures are then appended to each sample in the minibatch and fed into the generator, where they are used to introduce diversity in the generated samples.
* **Historical Averaging**, slows down the training of GANs when the model parameters are changing too quickly over time. To achieve this, the technique involves adding an additional term to the loss function that takes into account the previous configuration of the model parameters. This technique helps to stabilize the training process and prevent the model from overfitting to the data.
* **One-Sided Label Smoothing**, replaces the crisp binary labels assigned to real and fake samples with softer values that are slightly shifted towards the opposite class. This helps to prevent the discriminator from becoming overconfident about its decision and reduces the risk of the generator overfitting to the discriminator's decision boundary.
* **Virtual Batch Normalization (VBN)**, involves maintaining a separate reference batch that is used to normalize the activations of each layer during training. This allows for greater consistency in the normalization of the network's inputs, which can help to improve the stability and generalization of the model.
* **Adding Continuous Noises** can be added to the inputs of the discriminator during training. This causes the probability distribution to be more spread out and increases the chances of overlap between the real and synthesized data distributions. By doing so, the discriminator is forced to learn more about the variations in the data and is less likely to memorize specific examples.

Chapter 3

Problem Statement / Requirement Specifications

In this section, we discuss the problem statement and the requirements for the proposed system.

3.1 – Problem Statement

The problem addressed in this survey is the challenge of keeping up with the constantly evolving field of Generative Adversarial Networks (GANs) in deep learning, which have had a significant impact on society. With new research and advancements in GANs being published frequently, it is difficult for researchers to stay up-to-date with the latest architectures, optimization techniques, validation metrics, and application areas of GANs. The purpose of this survey is to present a detailed summary of GANs, assess the effectiveness of various versions of the model structure, and scrutinize the metrics and loss functions commonly used for evaluating GAN performance. The survey ultimately aims to guide future researchers in the field by summarizing the evolution and performance of the most promising GAN variants.

3.2 – Project Planning

* We studied various literature on **Generative Adversarial Networks** (GANs)
* We studied various problems based on GAN trainings and we observed problems like **MOD collapse**, **slow convergence rate** of discriminator and generator networks which lead to oscillation or divergence from the min-max point.
* We further studied about **Nash equilibrium** and **proximal convergence** of GANs for finding stable min-max points.
* We learnt about the mathematics leading to **Gradient Descent Ascent** and about the zero-sum game optimizations in non-convex contours of the loss functions.
* We further study about various evaluation metrics and various improved architectures.
* We summarise the performance of various GAN variants which aims future researchers in this field.

3.3 – Project Analysis

3.3.1 – Feasibility Analysis

Before proceeding with the project, a feasibility analysis was be conducted to assess if the project is feasible from technical, financial, and operational perspectives. The required technology and expertise are available to develop and implement the study on GANs and it does not outweigh the budget. It is compatible with existing tools and technologies used by the target audience, as well as the willingness of the target audience to participate in the survey.

3.3.2 – Risk Analysis

Risk analysis involves identifying potential risks that could negatively impact the project and developing a plan to mitigate those risks. Here are some potential risks for the project of conducting a survey on Generative Adversarial Networks (GANs):

* Research Complexity: The research on GANs is constantly evolving and expanding, making it difficult to keep up with the latest developments. This could result in incomplete information being presented in the survey.
* Technical Issues: The development of the survey platform and data analysis tools may encounter technical issues such as software bugs or compatibility issues, which could delay the project and impact the accuracy of the results.
* Availability of Resources: The project may face challenges in terms of access to funding, research materials, and expert knowledge, which could impact the quality and accuracy of the survey results.

To mitigate these risks, the project team implemented measures such as conducting frequent checks on the accuracy and completeness of the research, implementing rigorous testing and quality assurance procedures for the survey platform and data analysis tools, ensuring adequate resources are available for the project's success.

3.3.3 – Scope Analysis

The scope of the problem statement is to provide a comprehensive study on the latest architectures, optimization of loss functions, validation metrics, and application areas of Generative Adversarial Networks (GANs). The survey aims to evaluate the efficiency of different variants of the GAN model architecture and their best application areas. Additionally, the survey aims to analyse the different metrics for evaluating the performance of GANs and frequently used loss functions. The final objective of the survey is to provide a summary of the evolution and performance of GANs to guide future researchers in the field. The scope of the study does not include the development or implementation of a specific GAN system or model.

3.4 – System Design Constraints

3.4.1 – Software and Hardware Specifications

We utilized TensorFlow and Keras, which are popular machine learning libraries, to create and train neural networks. For cloud-based computation, we chose Google Colab as our platform, and imageio and matplotlib libraries were used for visualizing the data by creating graphs and gifs or videos. The implementation of the code was done on Google Colab on a Chrome Browser using the T4 GPU, which has 16 GB of GPU memory, 320 Turing Tensor Cores, and 2,560 CUDA cores. Its boost clock frequency can go up to 1590 MHz, and it supports CUDA compute capability 7.5.

3.4.2 – Dataset Used

MNIST is a commonly used dataset in the field of machine learning and computer vision. It consists of 70,000 grayscale images of handwritten digits from 0 to 9, each of size 28x28 pixels. The dataset is often used for image recognition tasks, particularly for training and evaluating models that can identify handwritten digits. MNIST has been widely used as a benchmark in the field and is commonly used to evaluate the performance of various machine learning algorithms.

3.4.3 – Experimental Setup

The experiment setup for this project involved utilizing the MNIST dataset to train and evaluate various GAN models. We used TensorFlow and Keras libraries for creating and training the models, and Google Colab as our platform for cloud-based computation. Our goal was to compare the performance of several different GAN variants, including DCGAN, WGAN, and CGAN, in terms of image generation quality and training efficiency.

Our experiment involved training each GAN variant with a fixed number of epochs and evaluating the generated images using various metrics such as Inception Score and Fréchet Inception Distance. We also experimented with different hyperparameters and loss functions to optimize the performance of the models.

Chapter 4

Implementation

4.1 – Model Architectures

4.1.1 - Deep Convolutional GAN (DCGAN)

The Deep Convolutional GAN (DCGAN) was proposed in 2015 as an improvement to the original GAN architecture proposed in 2014. DCGAN uses convolutional layers instead of fully connected layers, which have been traditionally used for computer vision tasks. Convolutional layers are used to extract important features from images and other types of data arranged in matrices. DCGAN also introduces other changes, such as replacing pooling layers with strided convolutions, using batch normalization layers, and using different activation functions for the hidden and output layers. These changes improve the stability and performance of GANs. The DCGAN paper presents a which aid in the understanding of GANs' learning methods. The DCGAN architecture has become a standard in GANs design and training, and its innovations are applied in most of the following GAN models.

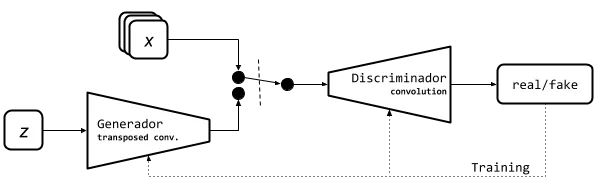


Figure 4.1: DCGAN Architecture

Loss Function:



Train Summary:

* Rescale the MNIST images between -1 and 1.
* Use transposed convolution in Generator with ReLU activation and BatchNormalisation. Tanh is the last activation.
* The Discriminator includes CNN and LeakyRELU action with last being Sigmoid.
* We use Binary Crossentropy Loss Function and Adam with lr = 0.0002 and beta\_1 = 0.5.

4.1.2 - Conditional GAN (CGAN)

In 2014, a new model called CGAN was proposed, which extends the traditional GAN architecture by adding a new latent class label c to the input. This additional information assists in classifying the generated data into distinct classes, resulting in the production of synthesized data that corresponds to the input class label. This approach is useful in problems that require data generation for various classes and has shown to prevent mode collapse effectively. However, its application to some problems is complicated due to the requirement of labelled datasets for training. Despite its simplicity, the CGAN model has had a significant impact on GAN models, and many variations have been developed since its introduction.



Figure 4.2: CGAN Architecture

Loss Function:

Cost function CGANs by fernanda rodríguez.

Train Summary:

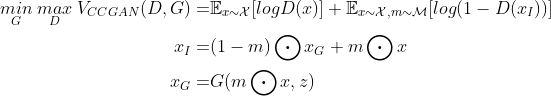
* Rescale the MNIST images between -1 and 1.
* Use transposed convolution in Generator with ReLU activation and BatchNormalisation. Tanh is the last activation.
* The Discriminator includes CNN and LeakyRELU action with last being Sigmoid.
* We use Binary Cross-entropy Loss Function and Adam with lr = 0.0002 and beta\_1 = 0.5.

4.1.3 – Context-Conditional GAN (CCGAN)

Context-Conditional Generative Adversarial Networks (CC-GANs) are a type of conditional GANs that focus on a different task compared to traditional GANs. CC-GANs aim to determine whether a specific region of an image is real or fake by looking at the surrounding context. The generator is trained to fill in the missing region of an image while being conditioned on the surrounding pixels. This is done by providing the generator with an image where a random patch has been masked out, and then training it to generate the missing patch while taking into account the surrounding pixels. The completed image is then passed into the Discriminator 𝐷 to determine whether the masked-out patch is real or fake. CC-GANs have a unique ability to fill in missing image patches and detect fake patches within a specific context. 

Figure 4.3: CCGAN Architecture

Loss Function:



Train Summary:

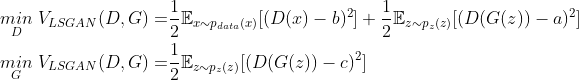
* Rescale the MNIST images between -1 and 1. Resize 32x32
* Generator is a U-net network. The input is the normal distribution z and the last activation is tanh.
* Discriminator is a CNN and have LeakyReLU activation. The last activation is a softmax.
* We use Categorical CrossEntropy and MSE Discriminator loss and the adversarial loss is MSE.
* We use Adam as optimiser with lr = 0.0002 and beta\_1 = 0.5

4.1.4 – Least Square GAN (LSGAN)

Least Squares Generative Adversarial Networks (LSGANs) is a type of generative adversarial network (GAN) that uses the least squares loss function instead of the binary cross-entropy loss function commonly used in traditional GANs. LSGANs aim to improve the stability and quality of generated samples by adopting a different approach to the loss function used in the Discriminator. 

Figure 4.4: LSGAN Architecture

Loss Function:



Train Summary:

* Rescale the MNIST images between -1 and 1.
* Generator is a simple fully connected neural network with Leaky ReLU activation and BatchNormalisation. The last activation is tanh.
* Discriminator is a simple fully connected neural network with Leaky ReLU activation. The last activation is a sigmoid.
* We use MSE as loss and Adam as optimiser with lr=0.0002 and beta\_1=0.5.

4.1.5 – Wasserstein GAN (WGAN)

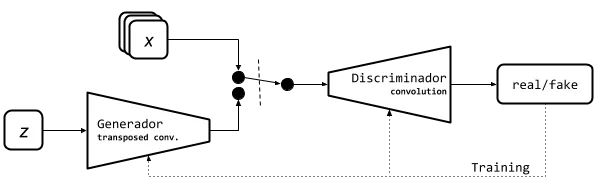
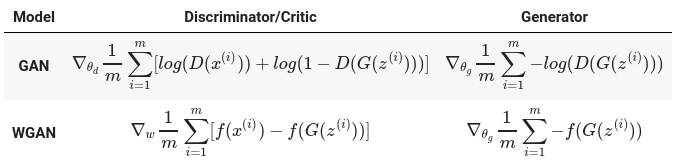
The WGAN architecture is a modification of traditional GANs that uses the Wasserstein-1 distance to measure the correlation between synthesized and real data distributions. This distance is based on the cost of transforming one distribution into another. To implement this new objective function, changes are made to the GAN architecture. The discriminator is now called the "critic" and is responsible for measuring the realness of an image. To keep the parameters in a compact space, weight clipping is used. Compared to traditional GANs, WGAN has better convergence, stability, and mode collapse avoidance, particularly in low-dimensional manifold distributions. The WGAN loss function measures the distance with the help quality of fake samples and converges to a minmax point. 

Figure 4.5: WGAN Architecture

Loss Function:



Train Summary:

* Rescale the MNIST images between -1 and 1.
* Generator uses transposed convolution with ReLU activation and BatchNormalisation. The last activation is tanh.
* Discriminator is a CNN with Leaky ReLU activation. There is no last activation function.
* We use Wasserstein loss as loss and RMSprop as optimiser with lr=0.00005.

4.2 – Observations (Loss per Epoch for first 100 epochs)

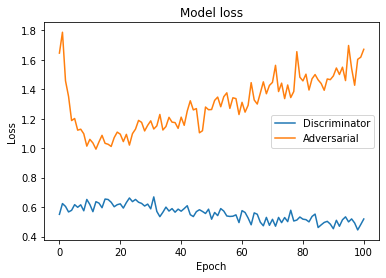


Figure 4.6: Original GAN on MNIST

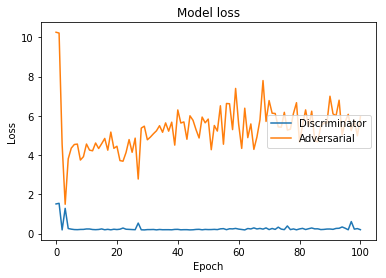


Figure 4.7: DCGAN on MNIST

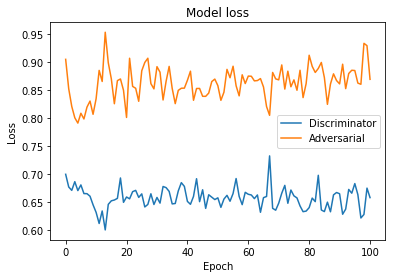


Figure 4.8: CGAN on MNIST

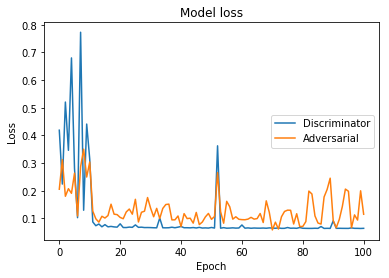


Figure 4.9: CCGAN on MNIST

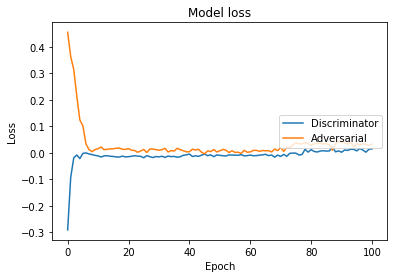


Figure 4.10: WGAN on MNIST

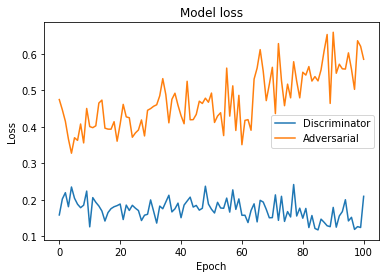


Figure 4.11: LSGAN on MNIST

4.3 – Result Analysis

Generative Adversarial Networks - GANs

|  |  |
| --- | --- |
| Epoch 00 | Epoch 100 |
| GAN with MNIST | GAN with MNIST |

Deep Convolutional Generative Adversarial Networks - DCGANs

|  |  |
| --- | --- |
| Epoch 00 | Epoch 100 |
| GAN with MNIST | GAN with MNIST |

Conditional Generative Adversarial Nets - CGANs

|  |  |
| --- | --- |
| Epoch 00 | Epoch 100 |
| CGAN with MNIST |  |

Context-Conditional Generative Adversarial Networks - CCGANs

|  |  |
| --- | --- |
| Epoch 00 | Epoch 100 |
| CGAN with MNIST | WGAN with MNIST |

Wasserstein Generative Adversarial Networks - WGANs

|  |  |
| --- | --- |
| Epoch 00 | Epoch 100 |
| WGAN with MNIST | WGAN with MNIST |

Least Squares General Adversarial Networks - LSGANs

|  |  |
| --- | --- |
| Epoch 00 | Epoch 100 |
| LSGAN with MNIST | LSGAN with MNIST |

4.4 – Quality Assurance

Our project has successfully met the necessary quality standards by adhering to industry best practices and guidelines. We have utilized appropriate methodologies, techniques, and tools to maintain quality throughout the project development lifecycle. Regular testing and validation have been conducted to verify that the system complies with the required specifications and delivers the anticipated results. Our team has also followed quality assurance processes to identify and address any defects or issues in the system. Additionally, we have kept the project documentation up-to-date and precise, reflecting the system's design and functionality. Proper communication and collaboration with stakeholders have been maintained to ensure on-time delivery and satisfaction with the final product.

Chapter 5

Standards Adopted

5.1 – Design Standards

The project's design adhered to the principles of modularity and extensibility. The system was structured in a layered architecture, with each layer fulfilling a specific function. The layers were devised with flexibility in mind, making it straightforward to modify or enhance them in the future.

5.2 – Coding Standards

The code was well-documented and follow standard coding practices. The code was written in a way that is easy to read and understand. The code followed a consistent coding style.

* The code was written as short as possible with industry-standard refactoring.
* Usage of appropriate naming conventions was followed.
* Code was properly indented with 4 spaces.
* A single function has carried out a single specific task

5.3 – Testing Standards

The system was tested thoroughly before deployment. The system is tested on a large dataset and should be able to handle different types of input images. The system should also be tested for different image processing tasks such as image super-resolution, style transfer, and image synthesis.

Chapter 6

Conclusion and Future Scope

6.1 – Conclusion

This report provides an overview of recent advancements in GANs, from their fundamental principles to the latest innovative architectures. The report categorizes the various problems that GANs can encounter and explains the most commonly used evaluation metrics. The report proposes a taxonomy for GAN variants, dividing them into two groups based on whether they focus on architecture optimization or objective function optimization. It is emphasized that these groups are not mutually exclusive and that each variant builds upon the progress of previous research. The report covers a range of topics related to GANs, including their applications, impact on society and industry, and the evolution of GAN architectures. The report also includes a quantitative analysis of the performance of different GAN models to provide a clear picture of their progress and effectiveness over time.

6.2 – Future Scope

The future scope of GANs is vast and promising. One potential area of research is the development of more efficient and effective training algorithms for GANs. This could involve exploring different optimization techniques, regularization methods, or training strategies to improve the stability and convergence of GANs.

Another area of research is the application of GANs in new domains, such as natural language processing, where GANs have already shown promising results. The development of GANs for multi-modal data, such as video or audio, is also an active area of research.

Additionally, there is a growing interest in developing GANs that can operate in more complex and dynamic environments, such as real-time video processing or robotic control. These applications require GANs that can adapt quickly to changing environments and generate high-quality outputs in real-time.

Overall, the future of GANs is exciting, and the potential applications of this technology are vast. Continued research and development in this field are likely to lead to significant advancements in machine learning, computer vision, and other areas of artificial intelligence.

References

[1] Generative Adversarial Nets. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio.

[2] UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. Alec Radford & Luke Metz, Soumith Chintala.

[3] Conditional Generative Adversarial Nets. Mehdi Mirza, Simon Osindero.

[4] Wasserstein GAN. Martin Arjovsky, Soumith Chintala, and L´eon Bottou

[5] Least Squares Generative Adversarial Networks. Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley

[6] COMPARISON OF GENERATIVE ADVERSARIAL NETWORKS IN MEDICAL IMAGING APPLICATIONS. MR to CT image synthesis. Christoffer Edlund.

[7] Continuous Conditional Generative Adversarial Networks: Novel Empirical Losses and Label Input Mechanisms. Xin Ding, Yongwei Wang, Zuheng Xu, William J. Welch, and Z. Jane Wang.

[8] A survey on GANs for computer vision: Recent research, analysis and taxonomy. Guillermo Iglesiasa, Edgar Talaveraa, Alberto D´ıaz-Alvarez.

[9] From GAN to WGAN. Lilian Weng.

[10] Conditional Image Synthesis with Auxiliary Classifier GANs. Augustus Odena, Christopher Olah, Jonathon Shlens.

**SAMPLE INDIVIDUAL CONTRIBUTION REPORT:**

**ON A COMPARATIVE ANALYSIS OF VARIOUS GAN MODELS**

Sandeep Kumar Swain

20051025

**Abstract:** The aim of the project is to provide a comprehensive overview of Generative Adversarial Networks (GANs), including their latest architectures, validation metrics, and applications. The project's objective is to evaluate the effectiveness of different GAN variants, identify the best application areas, and summarize the evolution and performance of GANs. The project seeks to provide guidance for future researchers in the field and serve as a reference for those seeking to understand GANs' capabilities and limitations.

**Individual contribution and findings:**

In this project, I delved into the mathematical foundations of Generative Adversarial Networks (GANs) and their associated loss functions. Specifically, I studied the contours of the loss functions used in GANs, including the generator and discriminator loss, and how they affect the training process. I also explored various optimization techniques used to train GANs including change in optimization algorithms and the GAN architectures.

**Individual contribution to project report preparation:**

I contributed in the “basic concepts” chapter of the report and the mathematical equations involved in the report.

**Individual contribution for project presentation and demonstration:**

I provided the slides containing the mathematics and their justifications.

Full Signature of Supervisor:                                 Full signature of the student:

**SAMPLE INDIVIDUAL CONTRIBUTION REPORT:**

**ON A COMPARATIVE ANALYSIS OF VARIOUS GAN MODELS**

Shivam Mishra

20051028

**Abstract:** The aim of the project is to provide a comprehensive overview of Generative Adversarial Networks (GANs), including their latest architectures, validation metrics, and applications. The project's objective is to evaluate the effectiveness of different GAN variants, identify the best application areas, and summarize the evolution and performance of GANs. The project seeks to provide guidance for future researchers in the field and serve as a reference for those seeking to understand GANs' capabilities and limitations.

**Individual contribution and findings:**

In this project, I contributed to the coding part by implementing various GAN models using TensorFlow or Keras. To ensure that the models were properly implemented and trained, I carefully followed the model architecture and loss function specifications provided by the project team. I also fine-tuned the hyperparameters to optimize the training process and improve the performance of the models.

**Individual contribution to project report preparation:**

I contributed in the “implementation” chapter of the report and the diagrams related to the architecture along with the Training Summary.

**Individual contribution for project presentation and demonstration:**

I provided the slides containing the architecture diagrams and their layers.

Full Signature of Supervisor:                                 Full signature of the student:

**SAMPLE INDIVIDUAL CONTRIBUTION REPORT:**

**ON A COMPARATIVE ANALYSIS OF VARIOUS GAN MODELS**

Shivansh Maheswari

20051029

**Abstract:** The aim of the project is to provide a comprehensive overview of Generative Adversarial Networks (GANs), including their latest architectures, validation metrics, and applications. The project's objective is to evaluate the effectiveness of different GAN variants, identify the best application areas, and summarize the evolution and performance of GANs. The project seeks to provide guidance for future researchers in the field and serve as a reference for those seeking to understand GANs' capabilities and limitations.

**Individual contribution and findings:**

In this project, I implemented and trained six different Generative Adversarial Network (GAN) models on the MNIST dataset. These models included vanilla GAN, Deep Convolutional GAN (DCGAN), Conditional GAN (CGAN), Context-Condition GAN(CCGAN), Wasserstein GAN (WGAN), Least Squares GAN (LSGAN). To ensure that the models were properly trained and performed well, I chose appropriate hyperparameters for each model.

**Individual contribution to project report preparation:**

I contributed in the “observations and result analysis” chapter of the report. The outputs and the results involved in the report.

**Individual contribution for project presentation and demonstration:**

I provided the slides containing the outputs and the final results.

Full Signature of Supervisor:                                 Full signature of the student:

**SAMPLE INDIVIDUAL CONTRIBUTION REPORT:**

**ON A COMPARATIVE ANALYSIS OF VARIOUS GAN MODELS**

Pranshu Priyaranjan

20051018

**Abstract:** The aim of the project is to provide a comprehensive overview of Generative Adversarial Networks (GANs), including their latest architectures, validation metrics, and applications. The project's objective is to evaluate the effectiveness of different GAN variants, identify the best application areas, and summarize the evolution and performance of GANs. The project seeks to provide guidance for future researchers in the field and serve as a reference for those seeking to understand GANs' capabilities and limitations.

**Individual contribution and findings:**

In this project, my contributions included documentation of the project cycle, creation of a presentation summarizing the results, and evaluation of the models' performance. Through my efforts, I helped ensure that the project was well-documented and the models were performing effectively, which was critical to the success of the project.

**Individual contribution to project report preparation:**

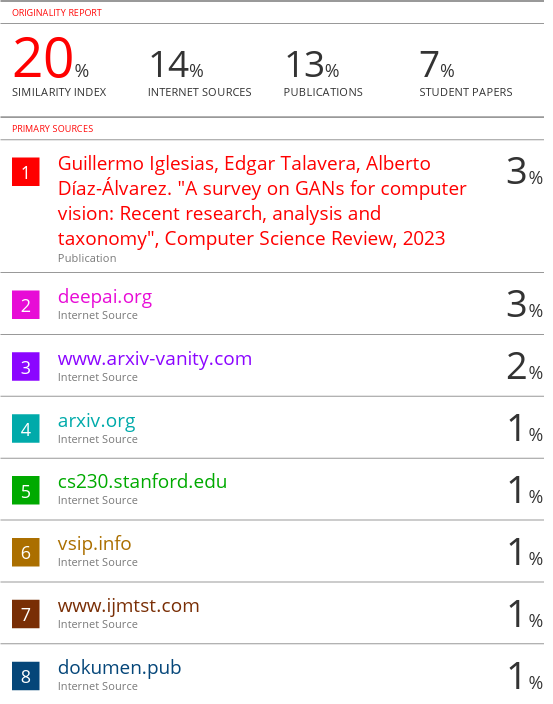
I contributed in the “overall theory” part of the report and its re-valuation. The aim was to make the documents as concise as possible for the new readers.

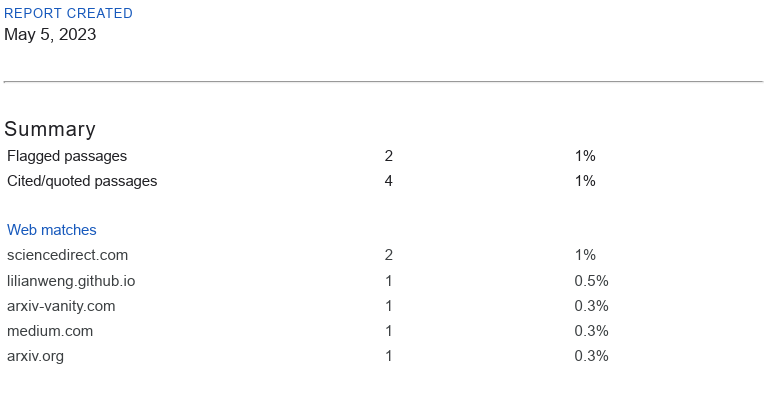
**Individual contribution for project presentation and demonstration:**

The overall slides were prepared by me taking the various inputs from the team members.

Full Signature of Supervisor:                                 Full signature of the student:

**TURNITIN PLAGIARISM REPORT**

****

**Google Classroom Plagiarism Report**