Study on Deep Generative Models
Generative Adversarial Networks

A FORWARD STUDY OF ANIME FACE GENERATION

VIVIAN SEDOV

Final Report
BSc (Hons) in Computer Science and Artificial Intelligence
SUPERVISED BY: DR YUNKUEN CHEUNG

DEPARTMENT OF COMPUTER SCIENCE
ROYAL HOLLOWAY, UNIVERSITY OF LONDON

March 2023

Contents

List of Figures

Abstract

Since the introduction of generative networks, image generation has made remarkable progress, particularly in the field of facial construction. Ian Goodfellow's introduction of Generative Adversarial Networks (GANs) [4] in 2014 opened up new possibilities for generating realistic images. In recent years, many attempts have been made to use GANs for generating anime faces. However, generating anime faces poses unique challenges due to their stylized and cartoon-like features. Unlike realistic human faces, anime faces often have exaggerated features, such as large eyes and bright-colored hair, that can be difficult to represent accurately using traditional image generation techniques. Additionally, anime artists may use a wide range of styles and techniques, which can make it challenging to develop a generalizable model for anime face generation.
To address these challenges, the proposition is to use label synthesis and feature labeling tools such as Illustration2Vec to identify and tag features of anime characters. This approach allows us to compile a more accurate and well-suited dataset for training GAN models. Through the use unsupervised learning techniques to apply GANs and evaluate the effectiveness of different generative models. By combining data and model perspectives, the project aims to quantitatively analyze facial production and generate high-quality anime faces that are faithful to their original style and characteristics [8].

The proposed study aims to generate anime faces using unsupervised learning with GANs and to compare the effectiveness of different generative models. This study will discuss the operation and architecture of several generative models. Including the Wasserstein Auxiliary Classifier GAN (WAC-GAN) [5], which demonstrates improved performance and stability in generating high-quality images, especially in the context of imbalanced datasets. With the effectivness of ACGans, this study with discuss the integration of Illustration2Vec with respect to Active label sythesis, and the application of Auxiliary Classifier Generative Adversarial Network (AC-GAN) [12]. The aim to achieve an automatic and credible process in generating anime-style images while maintaining their unique artistic characteristics [2] [19] [16].

Generative models, Unsupervised learning, Generative Adversarial Network, Machine Learning

March 21, 2023

1 Introduction

The fast growth of Japanese animation industry has resulted in the production of many animated shows that have piqued the attention of diverse audiences. Each anime has a unique trait that caters to a distinct users taste. Such that the idea of creating our custom ones come to mind. However, it takes tremendous efforts to master this skill, to draw and generate concise facial data. To bridge this gap, we have the principle of automatic generation of characters based on prior datasets. Due to the powerful ability of generative models, GANs have achieved great success in manipulating and generating images, with up-scaling and image generation being prime examples.

There are three essential requirements that must be met for the development of anime faces to ensure the success of this endeavour. I) Produced faces must be those of female and male faces, with a discernible difference between them; II) Each face must have unique traits that are sufficiently distinct from the initial training batch; III) The quality of the generated faces must be high[8]. To verify that these standards are adhered to, I will quantify the variety of the produced and input data using the following techniques: By having a suitable way to evaluate the performance of the produced faces, would imply the comparison between the generated photos towards the present data set. Various approaches, such as mean squared error or a similarity index based on the entropy of the produced pictures, may be used to quantify this. To confirm the image's viability, we may assess the quality of created pictures using the Inception Score *The Frechet Inception Distance (FID) : It measures the distance between two distributions, in this case the distribution of generated images from a GAN and the distribution of real images.*

Parametric models have shown great promise in the field of facial generation due to their ability to create new and unique faces by manipulating a set of parameters. However, there are inherent challenges in designing parametric models that can capture the complexities and diverse range of facial structures, especially in the context of anime faces. One significant challenge is the non-linear nature of facial structures, which can lead to inconsistencies and artefacts when generating new faces. Additionally, the high dimensionality of the feature space can make it difficult to control the generated faces' specific attributes, such as hair and eye color [9]. However, despite the availability of large datasets such as Danbooru2021, the inherent challenges of dataset dependability, quality, and biases still persist. Ensuring that the dataset is accurately labeled and free from erroneous or inconsistent information is critical, as GANs can be notoriously difficult to train. Another challenge that remains is the accurate generation of both male and female anime faces, as gender distinction is a crucial aspect of generating distinct and unique characters.

Various efforts have been made in the literature to generate anime facial data, with early studies by Alec Radford [14], Mattya [10], and Rezoolab [15] focusing on the key characteristics of anime face data. Subsequent projects such as IllustrationGAN [19] and AnimeGAN [6] have attempted to generate high-quality anime faces, but often report issues with fuzzy and deformed images, it is still difficult to develop a high standard anime faces for anime characters.

In this project, the proposal is a novel approach to overcome these limitations and generate high-quality anime faces by leveraging Illustration2Vec for active label synthesis and AC-GANs for image generation [16]. The use of Illustration2Vec allows us to automatically and accurately label the dataset with tags corresponding to style, gender, hair color, and eye color. These labeled features are essential for training AC-GANs, which in turn facilitate the generation of more diverse and high-quality anime faces while maintaining control over desired attributes.

By incorporating Illustration2Vec and AC-GANs in our approach, we aim to address the challenges faced by previous projects and generate visually appealing anime faces that respect the unique style and philosophy of anime characters. This combination of techniques not only helps to improve the overall quality of the generated images but also allows for more flexibility and control over the output, resulting in a more practical and versatile solution for anime face generation.

Keywords *Generative models, Unsupervised hyperref colorearning, Generative Adversarial Network, Machine Learning.*

2 Scope

2.1 Aim and Scope

The primary aim of this project is to develop a generative model capable of producing high-quality and unique anime character faces while addressing the inherent challenges and building upon the limitations of existing methods. This study will focus on exploring the potential of ACGAN and DCGAN architectures in generating distinct male and female faces with diverse traits and maintaining high image quality. Moreover, the application of Illustration2Vec will be investigated for accurately labeling the dataset and extracting semantic information from anime-style illustrations, which will facilitate generating images based on specific user requirements.

The objectives of this project are as follows:

- Examine the fundamental problems associated with GANs, such as mode collapse, convergence issues, and vanishing gradient, within the context of anime face generation, and explore potential solutions.
- Conduct a comparative analysis of different GAN architectures, ranging from basic GANs to more advanced variants like DCGANs and ACGANs, to assess their suitability for generating diverse and high-quality anime faces.
- Investigate the utility of Illustration2Vec for accurately labeling and extracting semantic information from the dataset, enabling the generation of anime faces based on specific user-defined attributes.
- Evaluate the performance of the developed model by quantifying the variety and quality of the generated faces, ensuring that they meet the essential requirements outlined in the introduction [8].

2.2 Data objective

Ideally in this study I want to seek to bridge the gap between the need for developing personalised virtual pictures through the principle of generating images based on personalised labels. Through leberaging the capabilities of Illustration2Vec [16] and AC-GANs / DCGANS [13]. In essence, a weakly supervised learning process using the aggregation of 2 core distinct learners, when being parsed into the generative network.

Such that the core objectives for the data it self are the following:

- Obtaining the ground truth for style labels: Eye colour, hair colour, and having a modular approach.
- Interpolating the image and parser to provide a more accurate representation of the image.

2.2.0.0.1 Pre-Processing As some datasets require customized images or pre-processed images based on their categorical information, it is essential to have more information about the images themselves. The core dataset used in this study has some labels, but it is necessary to consider additional labels and pre-trained models to expand on this information.

Experiments will be conducted on Illustration2Vec [19], a CNN-based technique for predicting tags for anime illustrations. Given an anime image, the network should predict the probability that it has generic anime features, referred to as tags (e.g., a smile, blue hair).

Although the core dataset I will be using has some labels, it is vital to have more information about the images it self. While the idea of manually annotating a ptoion of a gender label and then utilising it in conjuction with an incomplete labels sounds fruitful, it is also important to consider additional labels, and some pretrained models to expand on.

During the data pre-processing phase, the objectives are to:

- Determine the optimal way to represent and communicate the data to the generative network and the model
- Validate image quality
- Validate labels
- Parse forward to the generative adversarial network

Consequently, within the project's scope, I will strive to bridge the aforementioned disparities in a principled and organised way, enabling the user to easily design and produce pictures depending on particular image qualities. Such that this project aims to address the existing challenges with image labels and generate high-quality personalized virtual images.

2.3 Objectives - MileStones

2.3.1 Completed Objectives

- Gained an understanding of the mathematical principles behind GANs and the fundamental reasons why they are challenging to train.
- Investigated different GAN architectures by hyperparameter tuning and testing multiple network models, including DCGANs and MPL GAN (Multi Perceptron Layer).
- Created the initial anime dataset and applied a style network for feature extraction.

Upon the successful completion of these bjectives, this project will contribute to the development of a robus and effective Gan based solution for generating high quality, unique anime characters. The feats from this project will have future implications, particularly in the context of CGANs and a core future focus on ContraGans, which aim to enhance the controllability and interpretability of generated images. The insights gained from this project will pave the way for further exploration and refinement of GAN-based models, ultimately enabling more effective and efficient generation of diverse, high-quality, and user-specific content in various domains.

2.4 Related Literature

2.4.0.0.1 Generative Adversarial Networks (GANs) , introduced by Goodfellow et al [4]., demonstrate remarkable results in various generation tasks, such as image generation, image transfer, super-resolution, and more. The core concept of GANs involves training a generator and a discriminator model simultaneously. The discriminator's goal is to distinguish real examples, sampled from ground-truth images, from the samples produced by the generator. Conversely, the generator aims to create realistic samples that the discriminator cannot differentiate from ground-truth samples. This idea manifests as an adversarial loss applied to both the generator and discriminator during training, effectively encouraging the generator's outputs to resemble the original data distribution.

2.4.0.0.2 Conditional GANs (cGANs) , played a pivotal role in the early stages of this project, serving as a building block for generating images conditioned on specific input information. Mirza Osindero introduced cGANs, which incorporate conditional information, such as class labels, into both the generator and discriminator networks [11]. By conditioning the networks on additional input data, cGANs enable more controlled and targeted image generation, leading to better and more diverse results. The use of cGANs in this project allowed

for a more structured and meaningful approach to generating anime-style images, as it permitted the model to generate images based on specific input conditions. This technique improved the overall quality and relevance of the generated images while maintaining a strong foundation for further research and development.

2.4.0.0.3 Conditional Auxiliary Classifier GANs (AC-GANs) , Odena et al. introduced the concept of AC-GANs in their paper titled "Conditional Image Synthesis with Auxiliary Classifier GANs" [12]. AC-GANs integrate the ideas of cGANs [11] and auxiliary classifiers to generate images conditioned on class labels, which results in higher-quality image generation. The discriminator acts as an auxiliary classifier, predicting not only the realness of an image but also its class label. This dual role of the discriminator enables the generation of more accurate and diverse images, as the generator is encouraged to produce images that not only look realistic but also conform to the specified class label.

2.4.0.0.4 AnimeGAN , **AnimeGan & AnimeGANv2** are two GAN-based models specifically designed for generating anime-style images [7, 18]. These models inspired the development of this project's anime image generation framework. By analyzing the strengths and weaknesses of both AnimeGAN and AnimeGANv2, this research aims to improve upon existing techniques for generating anime-style images and address the limitations of these models.

2.4.0.0.5 Illustration2Vec , Illustration2Vec, proposed by Saito Matsumoto [16], is a model designed to extract semantic information from illustrations. It utilizes a deep convolutional neural network (CNN) to learn feature representations of illustrations, enabling various applications such as annotation, retrieval, and generation. By incorporating Illustration2Vec into the image generation process, it is possible to leverage the semantic information to create more accurate and diverse images. For example, combining Illustration2Vec with a GAN model allows the generator to produce images that not only look visually appealing but also possess meaningful semantic content. This can lead to more realistic and contextually appropriate image generation results.

Numerous GAN variants have been proposed for image generation. Radford et al. applied convolutional neural networks (CNNs) in GANs to generate images from latent vector inputs[14]. Instead of generating images from latent vectors, several methods utilize the same adversarial idea for generating images with more meaningful input. Mirza Osindero introduced Conditional Generative Adversarial Nets, which use image class labels as conditional input to generate specific MNIST numbers [11]. Reed et al. further employed encoded text as input to generate images that match textual descriptions. Odena et al. proposed ACGAN, which trains the discriminator as an auxiliary classifier to predict the input condition, instead of only feeding conditional information as input [12].

2.4.0.0.6 WCGANs Wasserstein GANs (WCGANs) were introduced by Arjovsky et al. [1] to address the issues of mode collapse and unstable training commonly associated with GANs. WCGANs employ a different loss function, the Wasserstein loss, which measures the Earth Mover's distance between the generated and real data distributions. This alternative loss function provides smoother gradients, leading to more stable training and improved convergence. Moreover, WCGANs incorporate a gradient penalty scheme that regularizes the discriminator's Lipschitz constraint, further stabilizing the training process and resulting in higher-quality generated images.

2.4.0.0.7 WACGANs Wang et al. proposed the Wasserstein Auxiliary Classifier GANs (WACGANs) in their paper "WACGAN: Wasserstein Auxiliary Classifier GAN for Imbalanced Data Generation" [3] [5]. WACGANs combine the advantages of WCGANs and ACGANs by integrating the Wasserstein loss function and gradient penalty scheme with the auxiliary classifier structure. This combination allows for more stable training and improved image generation, particularly for imbalanced datasets. WACGANs are capable of generating diverse and high-quality images conditioned on class labels, making them a relevant method in the context of anime-style image generation.

3 Background

3.1 Preliminaries and Analysiss

3.1.1 Background: Generative Adversarial Networks

Simply said, generative adversarial networks (GANs) are two neural network players competing to defeat one another. During this process, both neural networks gain a significant amount of information and produce more accurate findings, which we may exploit to create fresh synthetic data. This new information closely matches the original information exploited by the networks. GANs are being employed in the generation of pictures, sounds, and movies.

Generative Modeling is an unsupervised learning issue in machine learning that creates fresh content.The favourite example of Generative Adversarial Networks (which appears to be engrained at this point, comparable to Bob, Alice, and Eve for cryptography) is the situation of money counterfeiting. The Generator is meant to produce counterfeit currency, whilst the Discriminator is intended to discriminate between authentic and counterfeit dollars. As the Generator advances, the Discriminator must also progress. This implies a dual training strategy in which one model strives to outperform the other (i.e. via more learning) (i.e. through additional learning [**10.2312:egs.20191016**]

Particular a random vector/matrix, the generator offers a "fake" sample, and the discriminator attempts to detect whether a given sample is "genuine" (chosen from the training set) or "fake" (produced by the generator) (generated by the generator).

Training proceeds concurrently: the discriminator is taught for a few epochs, followed by the generator, etc. Thus, both the generator and the discriminator grow more adept at their respective vocations. GANs are extremely sensitive to hyperparameters, activation functions, and regularisation. They are notoriously tough to train.

3.1.2 Generator/Discriminator Network

The primary objective of the generator model within a generative adversarial network (GAN) is to generate synthetic data that can successfully deceive the discriminator. To achieve this, the generator typically begins with input noise, often Gaussian in nature, and transforms it into an image represented as a vector of pixel values. The generator's aim is to learn how to manipulate this noise to generate images that are classified as "real" by the discriminator.

During the generation process, loss is calculated whenever the discriminator accurately identifies an image as "fake." In parallel, the discriminator must learn to progressively distinguish between genuine and artificially generated images. If the discriminator fails to recognize a fake image, a negative loss is assigned to it. A crucial aspect of this approach is the concurrent training of both the generator and discriminator models, with each model continually adapting to the other's performance improvements.

A key aspect of GAN architecture is the simultaneous training of both the discriminator and generator models. As the discriminator becomes more proficient at identifying fake images, the generator must adapt and refine its synthetic data generation process to better deceive the discriminator. This adversarial relationship leads to a dynamic equilibrium, where both models continually improve until the generator produces highly realistic synthetic data.

In the provided example of a GAN architecture (Figure 1), a high-level overview of the generative network is depicted.

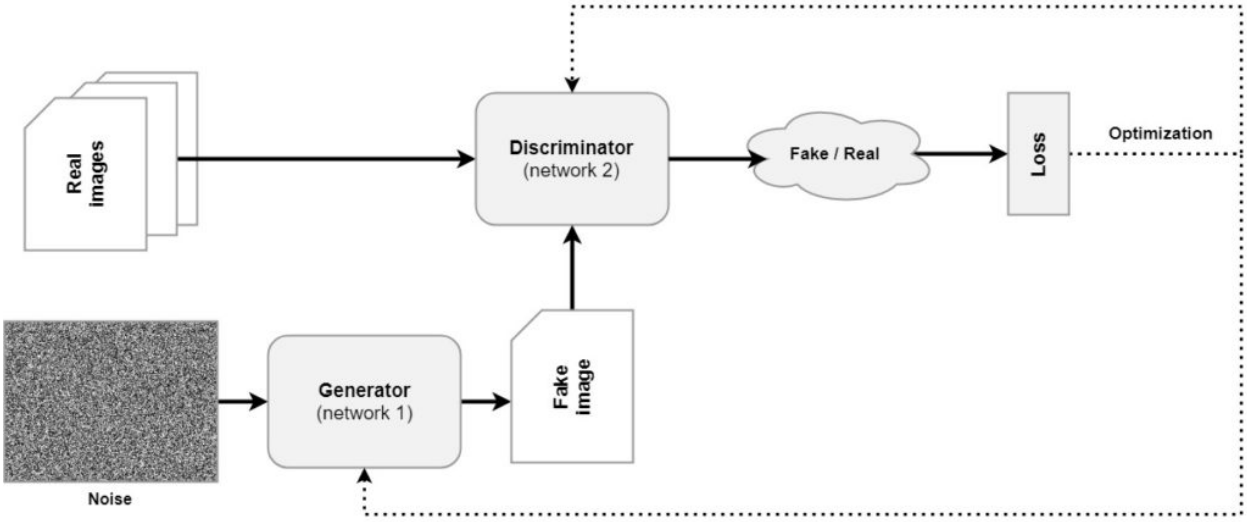


FIGURE 1 High level generative network.

Figure 1 GANs High Level Overview.
The core loss function for Vanilla GANs is given by:

$$\min_G \max_D \mathscr{L}(D,G) = \mathbb{E}_{x\sim P_{\text{data}}}[\log D(x)] + \mathbb{E}_{x\sim P_{\text{noise}}}[\log(1-D(G(z)))]$$

In this framework, the discriminator, denoted as D, aims to maximize the confidence scores it assigns to real samples while minimizing the confidence scores it assigns to the synthetic samples generated by the generator, G. Conversely, the generator's objective is to maximize the confidence scores that the discriminator assigns to its outputs, thereby attempting to deceive D more effectively.

The image illustrates the standard backpropagation process in which the generator receives feedback exclusively about its performance. The following section will delve into the fundamental mathematical principles underpinning GANs, providing a deeper understanding of their inner workings.

3.1.3 Core Mathematical Understanding of Vanilla GAN

A high-level overview offers a fundamental understanding, but to fully grasp the intricacies of generative networks, a deeper examination of their underlying mathematical principles is necessary. In this section, we will provide a thorough explanation and proof of the core mathematical concepts employed in the project. To achieve this, the objective must be explicitly stated in mathematical terms, guiding the proof technique and providing a target to focus on during the evaluation.

The goal of the Generator is to produce instances that are indistinguishable from real data. Mathematically, this corresponds to random variables with equal distribution (or in law), and their probability density functions (i.e., the probability measure generated by the random variable on its range) being identical:

$$p_G(x) = P_{data}(x)$$

. We will now outline the proof strategy: define an optimization problem in which the ideal solution is G satisfying $p_G(x) = P_{data}(x)$.

3.1.3.0.1 Creation of the loss function Drawing inspiration from the counterfeiting example, we first outline the rationale behind the formulation of the optimization problem. Firstly, we must ensure that the discriminator D can distinguish samples from $P_{data}(x)$:

$$E_{x\sim p_{data}(x)} \log(D(x))$$

Here, E denotes the expectation function (source). This expression enables the maximization of the expectation value, such that D predicts $D(x) = 1$ when the expectation is related to x for P_{data} of x.

The second core term is associated with the generator G. The Generator aims to deceive the discriminator, and this term arises from the negative loss:

$$E_{z\sim p_z(z)} \log(1 - D(G(z))).$$

A primary challenge in generative networks is the vanishing gradient problem. This issue arises when the discriminator's evaluation approaches $-\infty$, causing $D(G(z)) \approx 0$. The vanishing gradient problem is addressed in the algorithm detailed in Section (list section). To circumvent the vanishing gradient issue, gradient ascent is employed instead of gradient descent on the generated data, with the inverse log function used:

$$z \sim p_z(z) \log(D(G(z))).$$

Without this technique, vanishing gradient occurs, hence one of the primary objectives and purposes of this project is to fundamentally prevent this problem.

With that in mind, combining the two functions listed above provides the core loss function:

$$E_{x\sim p_{data}(x)} \log(D(x)) + E_{z\sim p_z(z)} \log(1 - D(G(z)))$$

This combination allows for the maximization of the discriminator with respect to G, ensuring the proper identification of real and fake data points.

The optimal discriminator can then be defined as follows:

$$V(G, D) := E_{x\sim p_{data}(x)} \log(D(x)) + E_{z\sim p_z(z)} \log(1 - D(G(z))).$$

We can further write $D^*_G = \text{argmax}_D V(G, D)$, where the Generator's objective is to reverse this process, attempting to minimize the prior equation when $D = D*_G$.

The original generative adversarial network paper by Ian Goodfellow [4] formulates the loss function as follows:

$$V(G, D) = E_{x\sim p_{data}(x)} \log(D(x)) + E_{z\sim p_z(z)} \log(1 - D(G(z))).$$

Due to the Ian Good fellows original paper relying on the equality of the following figure. This equality factor is derived from [Radon-Nikodym Theorem]. This theorem states that the derivative of the probability density function is equal to the probability density function of the derivative.

Another key factor in Vanilla Gans that is applied throughout all its varients is KL Divergence:

With this understanding in mind and how the loss function is derived, we can now move on to the next section and discuss the core mathematical principles of GANs, and the techniques that are used in different types of generative networks.

3.1.4 Machine Learning Technique

The core mathematical background explained earlier is applied to various machine learning techniques used in GANs. Such that it is the baseline and the core principle that gets expanded and modified upon. In this section, I will discuss core machine learning techniques that I have implemented and briefly talk about their differences. The core mathematical understanding and principle behind these models, and why they were selected will be referred in section 5.

The corresponding methods are required as a prerequisite towards the crucial understanding of the core process of this project. Everything listed below was used within the generation of the anime pictures.

3.1.4.1 GANs (GANs) / DCGANs (Deep Convolutional GANs)

DCGANs are a type of generative neural network that use deep convolutional layers to generate high-quality images. Some key techniques used in DCGANs include:

3.1.4.1.1 KL Divergence The Kullback-Leibler divergence is a measure of the difference between two probability distributions. It is used to measure the difference between the distribution of the real data and the distribution of the generated data. The KL divergence is calculated as follows: The KL divergence between two probability distributions P and Q is defined as:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} P(x) \log \frac{P(x)}{Q(x)} dx$$

where $P(x)$ and $Q(x)$ are probability density functions. This formula measures the amount of information lost when approximating P with Q. It is a non-negative value that is equal to zero only when P and Q are identical. In GANs, the generator network tries to produce samples that are similar to real data samples. The KL divergence is used to measure the difference between the distribution of generated samples from the generator and the distribution of real data samples.

In the context of this project: This was a crucial role toplay as all my models, were purely based on the original Vanilla Gan Varient, knowing this we can expand on the core ACGAN and WACGANs models.

For Vanilla Gans: the KL divergence is used as a measure between the distribution of the genereated data and its real data, this is likewise with deep convolutional layers, like dcgans. Leading into ACGANs, we base the KL Divergence in the context of the classifier.The generator produces not only the data samples but also a class label for each sample. The KL divergence is used as a measure of the difference between the class distribution of the real data and the generated data. The generator aims to minimize the KL divergence while the discriminator tries to maximize it.

3.1.4.1.2 Convolutional layers DCGANs are designed to extract features from the input data, typically images. These layers use filters or kernels that slide across the image and convolve with the pixel values to produce feature maps. The size of the filters can be chosen based on the specific problem, and the number of filters can be increased to capture more complex features in the input data. The following figure represents a convolutional layer:

3.1.4.1.3 Transposed convolutional layers on the other hand, are used to "upsample" the feature maps and generate larger outputs. These layers can be seen as the reverse operation of convolutional layers, and they are used to increase the resolution of the generated images.

3.1.4.1.4 Batch normalization Batch normalization is another technique used in DCGANs, which normalizes the inputs to each layer of the network. This can help to stabilize training and improve convergence, especially for deep networks. The normalization is performed over a batch of samples, and the resulting mean and variance are used to transform the inputs to each layer.

3.1.4.1.5 LeakyReLU activations are a type of activation function that are used in DCGANs to introduce nonlinearity into the network. LeakyReLU allows some negative values to pass through the network, which can improve the performance of the model. The function can be defined as:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \text{otherwise} & \alpha x, \end{cases}$$

where α is a small positive constant, typically set to 0.2.

3.1.4.2 Auxiliary Classifier GANs (ACGANs)

AC-GANs are a variation of DCGANs that include an auxiliary classifier, which allows the network to generate images conditioned on a specific class label. Some key techniques used in AC-GANs include: *This is further explain in section* **??**

3.1.4.2.6 One-hot encoding : This technique is used to represent the class labels as binary vectors, which can be fed into the network as additional inputs. One-hot encoding allows the class labels to be easily incorporated into the network architecture and helps to ensure that the labels are represented in a consistent and meaningful way.

Mathematically, one-hot encoding can be defined as follows: Let y_i be a class label for a given sample, where i is the index of the class label. The one-hot encoding of y_i is a binary vector $\mathbf{y}$ of length K, where K is the total number of classes. The vector $\mathbf{y}$ has all zero elements except for the element at position i, which is set to one. Softmax activation: This activation function is used to compute the probabilities of each class label, based on the output of the auxiliary classifier.

3.1.4.2.7 Softmax Activation Function Softmax is used to compute the probabilities of each class label based on the output of the auxiliary classifier. The softmax function can be defined as follows:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where z_i is the output of the auxiliary classifier for class i, and K is the total number of classes. The softmax function ensures that the predicted class probabilities sum to one, and it is commonly used in multi-class classification tasks.

3.1.4.2.8 Cross-entropy loss function as the objective function for the auxiliary classifier. Cross-entropy loss measures the difference between the predicted class probabilities and the true labels, and it is commonly used in classification tasks. The cross-entropy loss can be defined as follows:

$$\text{CE}(y, \hat{y}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i$$

where y is the true class label, $\hat{y}$ is the predicted class probabilities, and K is the total number of classes.

These techniques allow AC-GANs to generate images that are conditioned on a specific class label, which can be useful for a variety of tasks such as image synthesis and image manipulation. By incorporating an auxiliary classifier into the network architecture, AC-GANs are able to generate more diverse and meaningful images that are tailored to specific classes or categories.

3.1.4.3 WAC-GANs

: WAC-GANs, or Wasserstein Auxiliary Classifier Generative Adversarial Networks, are a variant of AC-GANs (Auxiliary Classifier Generative Adversarial Networks) that incorporate gradient penalty to improve training stability and performance. AC-GANs are a class of GANs that utilize conditional information during training to generate more specific and diverse data samples. WAC-GANs build upon this concept by integrating the Wasserstein distance metric for more stable training. WAC-GANs lies in the usage of gradient penalty, which addresses the issue of vanishing gradients in GAN training. By adding a penalty term to the discriminator's loss function, the model is encouraged to have smoother gradients, thus preventing the undesirable behavior of vanishing gradients. This leads to more stable and efficient training, ultimately resulting in the generation of higher-quality and diverse data samples.

Gradient penalty (GP) is a technique used to stabilize the training of GANs by enforcing a constraint on the gradients of the discriminator's loss function with respect to its input. It helps to tackle issues like vanishing gradients and mode collapse, which can result in poor quality of generated samples.

3.1.5 Mathematical Reasoning

Understanding the proof and theory behind GANs is necessary for future research, since it helps to confirm that the given solution is right and legitimate. In addition, it aids in identifying any possible problems or difficulties that may occur as a result of the suggested solution. In addition, knowing the evidence behind GANs allows academics and practitioners to better comprehend the model's basic concepts and logic, which may be leveraged to enhance its performance and precision. Understanding the evidence behind GANs may aid in the identification of possible applications and use cases for the model.

3.2 Challenges in GANs

GANs, while powerful in generating realistic images, face several challenges in their training process, such as mode collapse and training instability. In this section, It will briefly discuss these issues and how they relate to different GAN architectures. A comprehensive analysis of these problems and their corresponding solutions can be found in Section 5.1

3.2.1 Mode Collapse and Architectural Variations

GANs can suffer from mode collapse, where the generator focuses on specific modes of the target data distribution, neglecting its diversity. While DCGANs introduced architectural guidelines that improved training stability, they did not directly address mode collapse, leaving the issue unresolved.

ACGANs attempted to mitigate mode collapse by incorporating an auxiliary classifier into the discriminator, promoting the generation of diverse images across different classes. However, ACGANs may still experience mode collapse within individual classes and have unstable training in the early stages.

WACGANs combined the benefits of WGANs and ACGANs to more effectively tackle mode collapse. By employing the Wasserstein distance as the loss function and introducing a gradient penalty scheme, WGANs improved training stability and helped prevent mode collapse. Integrating these improvements with the auxiliary classifier concept from ACGANs, WACGANs offered a more robust solution for generating diverse images and mitigating mode collapse.

FIGURE 2 140 epochs

FIGURE 3 141 epochs

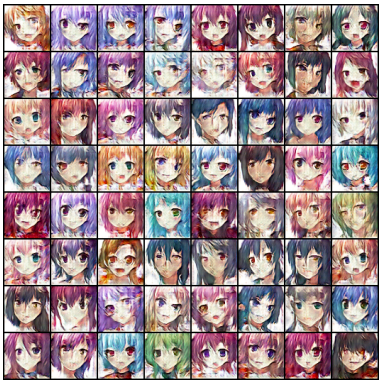


FIGURE 4 142 epochs

FIGURE 5 143 epochs

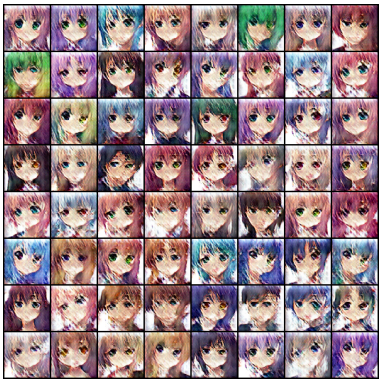


FIGURE 6 144 epochs

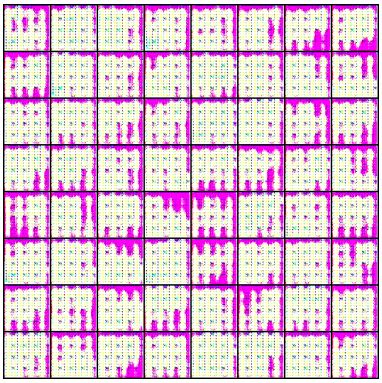


FIGURE 7 146 epochs

FIGURE 8 Images at different epochs 140 - 146

3.2.1.0.1 Why does it occur Mode collpase in this instance, occours for most models that do not have enough data, which in most cases is the issue. From a mathematical View mode collapse occurs from the following reason :

In the max-min solution, represented as $G^* = \min_{\theta_G} \max_{\theta_D} V(G,D)$, the optimal generator G generates samples from the data distribution. However, in the min-max solution, represented as $G^= \max_{\theta_D} \min_{\theta_G} V(G,D)$, the generator maps every z value to a single x coordinate that the discriminator believes to be real rather than fake. Simultaneous gradient descent does not provide a clear advantage for either the min-max or max-min solutions. Figure 8 shows the issue of both non convergence and mode collapse, most cases mode collapse leads into poor convergence issues. This example presents it self showing the issues with poor data and this was trained through a vanilla ACGAN.

Often, mode collapse is the result of inadequate generalisation. There are two sorts of mode collapse: (1) the majority of modes from the input data are missing from the produced data, and (2) G only learns a selection of specific modes. Many GANs versions, such as altering D's goal [17] and G's objective, have been suggested in response to the mode collapse issue, which may be exacerbated by an unsuitable objective function. In these forms, G is demonstrated to be capable of learning the whole data distribution at equilibrium, although convergence is difficult to achieve in practise.

One of the reasons for why I decided to use an Auxiliary generative network, and the combination of the Wasserstein allowed me to remove the issues that arise from mode collapse as present above. WACGANs tackle mode collapse by employing the Wasserstein distance as the loss function and introducing a gradient penalty scheme, one of the core issues that I had observed while training was the dimming of images, which is briefly mentioned, at section 3.3.2.

3.2.2 Training Instability

Training instability is a challenge that spans various GAN architectures. DCGANs improved stability through architectural guidelines and normalization techniques, but oscillations or divergence in training may still occur. ACGANs also faced training instability, particularly in the early stages, due to the added complexity of the auxiliary classification task.

WACGANs addressed training instability by incorporating the improvements from WGANs. The Wasserstein distance and gradient penalty scheme provided smoother loss landscapes and better convergence properties, leading to improved stability in the training process. Combining these enhancements with the auxiliary classifier from ACGANs, WACGANs presented a more stable and robust solution for training generative models and producing higher-quality generated images.

3.2.3 Vanishing Gradient Problem and Architectural Variations

The vanishing gradient problem impacts GAN training and the quality of generated images. Gradients flowing back to the generator during backpropagation may become very small, hindering the generator's ability to learn and update its parameters.

DCGANs mitigated the vanishing gradient problem to some extent through architectural guidelines and normalization techniques, ensuring a more stable gradient flow. However, the issue might still persist in certain situations. ACGANs, while encouraging diverse image generation, did not specifically address the vanishing gradient problem and may even exacerbate it under certain circumstances.

WACGANs, adopting the Wasserstein distance and gradient penalty scheme from WGANs, offered a more effective solution to the vanishing gradient problem. These enhancements ensured a more stable gradient flow during training, allowing WACGANs to provide a more robust approach for overcoming the vanishing gradient problem and generating high-quality images.

3.2.4 Lack of Diversity in Generated Images and Architectural Variations

Lack of diversity in generated images is a significant challenge for GANs, as it restricts the generator's ability to produce a wide variety of unique and realistic images that capture the variations present in the target data

distribution. In this section, I will discuss how different GAN architectures attempt to address this issue and their effectiveness in improving the diversity of generated images.

While DCGANs play a vital role here, they are not the key to producing diverse images, as presented as one of my key / core goals for this project. DCGANs provide the guidlines and the normalization techniques that improved training stability, but at the cost and at the fact that they did not inherently address teh lack of diverse images that are generated.

Moving forward With ACGANs, this aims on the principle of improving the diverse images that are generated. Due to its incorporation in auxiliary classifiers into the discriminator. This classifier guides teh generator to produce iamges with specific attributes, thus encouraging the generator to create images across different classes. While ACGANs can promote diversity across different classes, they may still struggle with achieving sufficient diversity within individual classes. Additionally, the introduction of the auxiliary classification task may increase the complexity of the training process, which can impact overall GAN performance.

This principles leads to WACGANs. By combining the benefits of WGANs and ACGANs, WACGANs more effectively address the lack of diversity in generated images. WGANs employ the Wasserstein distance and gradient penalty scheme to improve training stability and prevent mode collapse, which is closely related to the diversity issue. Integrating these improvements from WGANs with the auxiliary classifier concept from ACGANs, WACGANs offer a more robust solution for generating diverse and high-quality images. The WACGAN architecture allows for improved handling of both inter-class and intra-class diversity, leading to better performance compared to DCGANs and ACGANs.

3.3 GAN Performance: Dataset Size and Generator Strength

In some cases, GANs might generate images that appear darker than the training samples. One possible explanation for this behavior is the architecture of the generator itself. When training GANs, it is crucial to consider the overall quality of generated images, as well as the dataset size and the strength of the generator.

3.3.1 Dataset Size and Its Impact on GAN Performance

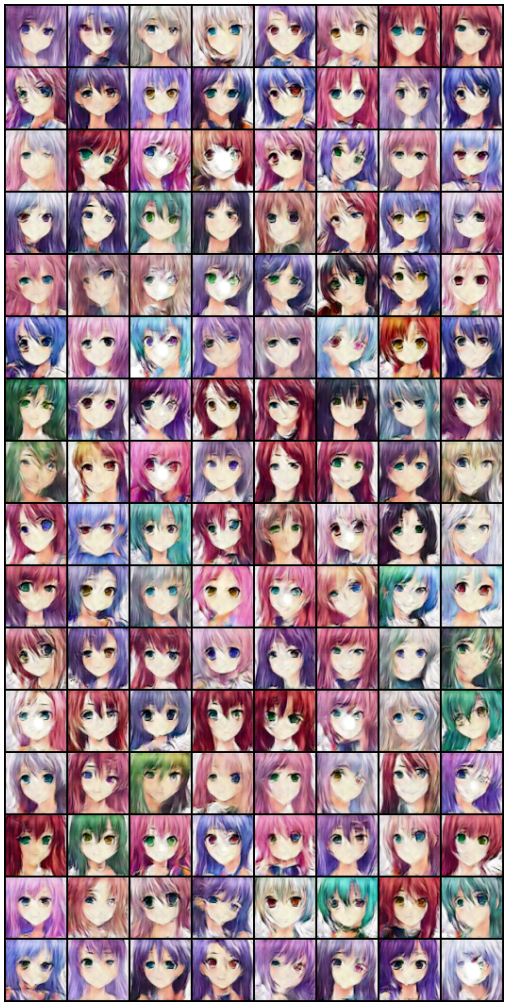
A large and diverse dataset is essential for training GANs effectively. The size of the dataset can significantly impact the performance of the generator, as well as the stability and convergence of the training process. Some of the ways in which dataset size affects GAN performance include:

3.3.1.0.1 Generalization : A larger dataset provides more diverse examples for the generator to learn from, Generalization is a crucial aspect of training generative models, as it ensures that the model can generate diverse and realistic images that capture the underlying data distribution. When a model generalizes well, it can create images that are not limited to reproducing specific examples from the training set, but instead encompass a broad range of variations present in the data. A great example is the following comparison, between two different datasets, one which is of 100 k images and the second of just under 20 k.

Overfitting is one of the fundamental problems associated with generalisation, and it is one of the primary obstacles in training generative networks. Overfitting happens when a model learns to replicate particular samples from the training set rather than capturing the underlying data distribution. This problem is particularly prevalent when training on smaller datasets since the model has fewer instances from which to learn. In generative models, overfitting may result in visuals that lack variety and closely mimic the training data, preventing them from generalising to unforeseen changes.

There are several strategies to avoid this, one of which is the concept of regularisation, which WCGAN implements. In most circumstances, this helps generalise the generative network since it inhibits overfitting by force. During training, dropout and weight decay may be used to avoid the model from depending excessively on a single input feature. In addition, architectural decisions such as batch normalisation may assist the model in discovering more stable and generalizable characteristics.

Furthermore Model Complexity is core, as balancing model complexity is a crucial part for achieving good generalization in generative networks, more over in the case of generating anime faces, mainly due to the complex nature of how they are structured. Such that if the model is too complex it may overfit the training data



and struggle to generate diverse images. On the other hand, if the model is too simple it may not be capable of learning the intricate patterns it presents within the data distribution. A well designed model is key for generating correct images. Which is why, I have a multitude of networks / Models to test from to see which one would evaluate the best.

3.3.1.0.2 Training stability : The size of the dataset used to train GANs plays a critical role in determining the performance and stability of the training process. A larger dataset provides a more diverse and comprehensive representation of the target distribution, reducing the likelihood of mode collapse and encouraging the generator to explore different modes present in the target data.

Training stability relies on 3 core components, while testing and creating this project such that :

- Larger datasets: A larger dataset exposes the generator to a greater variety of variables within the target distribution. This exposure enables the creator to learn and develop different outputs that span a variety of distribution channels. As a result, mode collapse is less likely to occur, since the generator is prevented from concentrating on a single mode or a small subset of modes. In this circumstance, the training process is more stable, and the picture generator is more suited to produce various, high-quality images. While this is one of the primary difficulties, training is exponentially increasing. It is consistent with the original nature of gans requiring considerable training time. In the instance of anime faces, it took me less than one week to correctly train one of my primary models.

- Comparatively, smaller datasets may lack adequate variety and richness, resulting in an insufficient representation of the intended data distribution. When trained on such datasets, the generator may have difficulty capturing the whole range of variability included in the target distribution. This restriction may enhance the possibility of mode collapse, since the generator is more likely to concentrate on a few dominating modes rather than explore the full distribution. Thus, the generator may provide outputs with restricted variance and lower quality, resulting in an unstable training process and inferior performance. Hence, if you compare Figure: 8 to Figure: **??**, you can notice a distinct change. One of my personal findings from this study is that about 70% of the key difficulties that one may have with generative networks may be fixed by expanding the data. While, as indicated, training and preprocessing would increase, it took me less than a day to adequately preprocess the data using illustrationtovec when I switched to 100k data. Further more, if the dataset is not viable / large enough you will occour Vanishing Gradient problem as refered at Section 3.3.1.0.3.

3.3.1.0.3 Vanishing gradient problem The vanishing gradient problem is a significant challenge in training deep learning models, including GANs, and it can be exacerbated by insufficient dataset size. When the dataset is not large or diverse enough, the generator struggles to learn and update its parameters due to the small gradients during backpropagation. This issue can negatively impact the quality of the generated images and lead to mode collapse, as illustrated in 8. Apart from dataset size, several other factors can contribute to the vanishing gradient problem in GANs:

- Network architecture
- Normalization techniques
- Loss function

Evaluating and comparing the performance of generative models is a non-trivial task due to the absence of a clear objective metric. Traditional evaluation metrics like accuracy or mean squared error are not directly applicable to generative models, as they focus on generating realistic samples rather than predicting specific outcomes. Researchers have proposed various evaluation metrics, such as Inception Score (IS) and Fréchet Inception Distance (FID), but there is no consensus on the best metric for all scenarios. This issue makes it challenging to compare different generative models and assess their performance objectively.

3.3.2 Generator Strength and Its Impact on GAN Performance

The strength of the generator is another critical factor that influences GAN performance. A strong generator is capable of learning complex patterns in the data distribution and generating high-quality images. Some aspects to consider when evaluating the strength of a generator include:

3.3.2.0.1 Architecture : The architecture of the generator plays a crucial role in determining its strength. A well-designed generator should include a combination of upsampling, convolutional, and normalization layers that facilitate the learning process. The use of transposed convolutions alone may not provide sufficient learnability, as they primarily serve as a heuristic for upsampling. To improve the generator's architecture, a process was implemented to add extra layers to the generator. These additional layers aim to increase the capacity of the generator, enabling it to capture more complex patterns and produce higher-quality images. By enhancing the architecture with more layers, the generator can benefit from increased learnability, making it better equipped to generate diverse and realistic images. This further lead to the idea presented in section 3.3.2 that the principle of darkended images is a fundemental flaw due to a poor architecture.

4 Feature Extraction/PreProcessing

As stated in Section **??**, One of the core principle is to determinal an optimal way to represent and communicate the dta, and to finally validate the data, in this subsection I will explore the core steps taken to use illustration2vec, and how it has evaluated to generative abilities, and creating labels and more.

4.1 Datasets

Image dataset quality is widely recognized as a critical factor for the success of image generation. Web services hosting image boards, such as Danbooru and Safebooru, are known for their extensive image collections. A striking contrast in the quality of generated images can be observed when comparing the images in Section 8 with those in Section **??**. During the first term of the project, I experimented with GANs using cats, dogs, and human datasets to evaluate their potential in generating viable images. The primary objective was to understand the fundamental principles of GANs, identify their limitations, and address the issues that might arise during the training process. The initial results provided valuable insights into the workings of GANs and helped shape the direction of the project moving forward listed examples of these will be presented in 5.0.3.

In the second term, I shifted my focus towards the Danbooru dataset. Although this dataset contained a vast collection of high-quality images, it proved to be computationally expensive to work with. Processing such a large dataset required significant resources and posed challenges in terms of time and cost efficiency. To overcome these hurdles, I considered using a pretrained dataset that was readily available in the project repository.

The inspiration for this project stemmed from the idea that, given the vast array of datasets and images available, there might be a domain issue concerning quality. As the project lead, it is my responsibility to address these concerns. It is evident that there is a considerable disparity in quality between images of anime faces and those of real people. To mitigate this issue, the data was cleaned and a custom label for *High Quality* images was introduced to ensure more consistent and reliable outcomes in the generated images.

4.1.1 Datasets used

4.1.1.1 MNIST

The MNIST dataset, which consists of 70,000 handwritten digits, is widely used as a benchmark for image recognition tasks. It was one of the initial datasets used in the first term to gain a deeper understanding of the working principles of GANs. By employing DCGANs with the MNIST dataset, I was able to identify the limitations and challenges associated with GAN training. Furthermore, the experience allowed me to comprehend the fundamental differences between generating grayscale images (MNIST) and more complex RGB images (cats, dogs, and humans).

4.1.1.1.1 Cats, Dogs, Human Datasets In addition to the MNIST dataset, I experimented with cats, dogs, and human datasets during the first term. These datasets were selected to investigate the capabilities of GANs in generating more intricate and diverse images. The primary goal of these experiments was to identify the issues and limitations that might arise when working with datasets containing a wide range of features and colors. The insights gained from these experiments played a vital role in shaping the direction of the project during the second term, as discussed in Section **??**.

4.1.1.2 Danbooru2021

The Danbooru2021 dataset, a large collection of high-quality images sourced from the Danbooru image board, was chosen as the primary dataset for the second term. However, processing such a vast dataset proved to be computationally expensive, requiring significant resources and time. Consequently, I decided to implement a feature extraction process to make the dataset more manageable.

Feature extraction involved manually assigning labels to each image, a labor-intensive task that demanded a considerable amount of time and effort. Despite the challenges, this process was crucial in ensuring a consistent

quality across the dataset. It enabled the identification and removal of low-quality images, as well as the introduction of a custom label for *High Quality* images. This approach not only improved the overall quality of the dataset but also enhanced the reliability and accuracy of the generated images.

4.1.1.3 Pretrained Danbooru2021+

To address the computational challenges associated with the raw Danbooru2021 dataset, I opted to use a pretrained version of the dataset, referred to as Pretrained Danbooru2021+. This dataset, available in the project repository, had already undergone feature extraction and preprocessing, allowing me to bypass the resource-intensive tasks while still benefiting from the rich collection of images.

By utilizing the pretrained dataset, I was able to focus on refining the GAN training process and improving the quality of the generated images. This decision led to more consistent and accurate results, ultimately contributing to the success of the project

4.2 Visualization of the Data

This on its own is an interesting prospective as for Visualization of the data, or the information that I am giong to provided, utalised the following machine learning methods:

- KMeans or MiniBatchKMeans clustering: This is a technique used for partitioning a dataset into k clusters, where k is a user-defined parameter. The algorithm works by iteratively assigning each data point to the cluster whose centroid is closest to it, and then recomputing the centroids of each cluster. KMeans is a popular clustering algorithm, but it can be slow and memory-intensive on large datasets. MiniBatchKMeans is a variation of KMeans that is faster and more memory-efficient, but may produce slightly different results.

Under Kmeans my memory maxed out, which lead me to look for alternative sources to represent the images. Such that referring from Machine Learning lectures; found that MiniBatchKmeans would be the most ideal and memory efficient method to generate a cluster like image form.

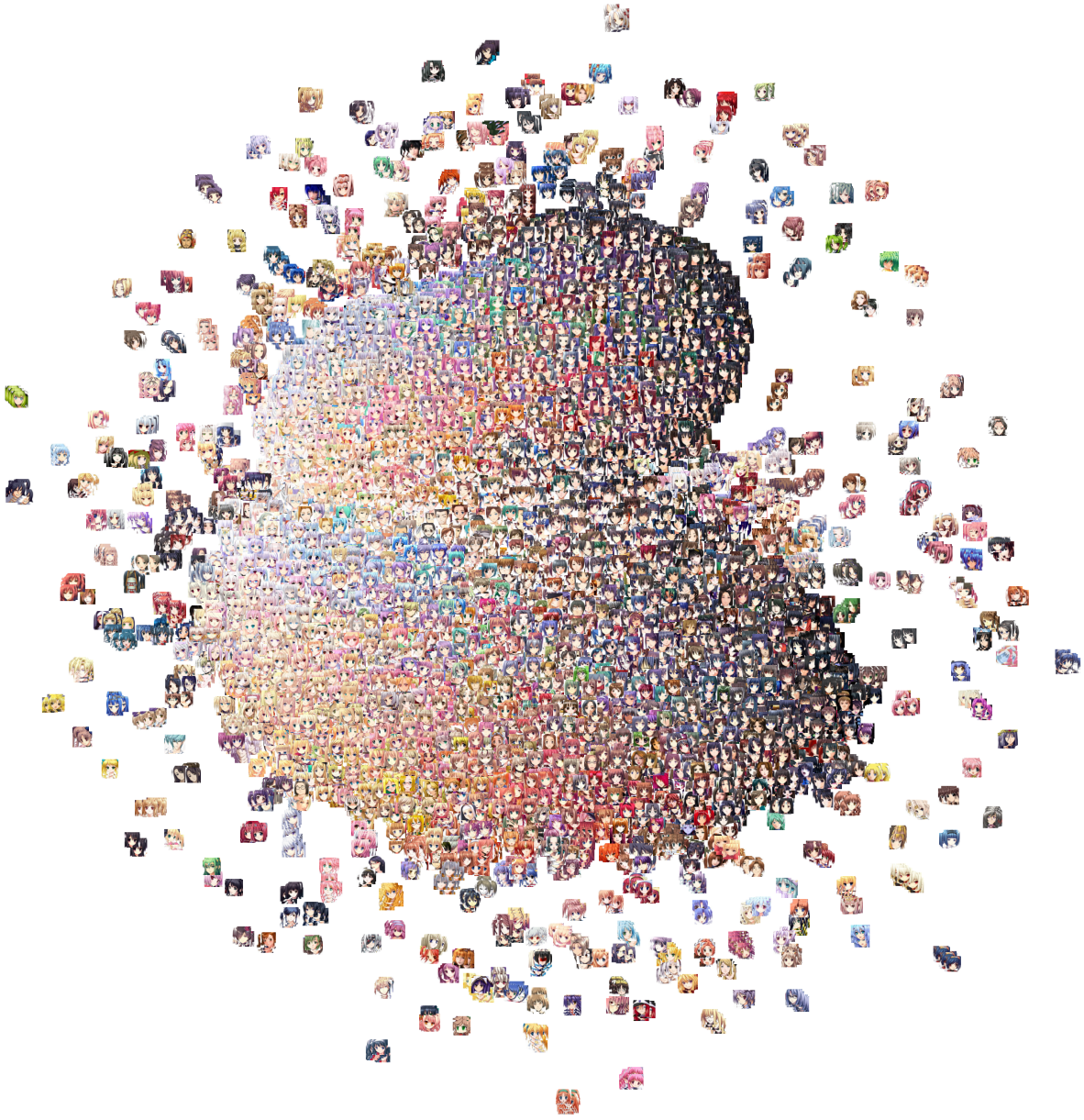


FIGURE 9 Cluster of images / Dataset used using umap

This dataset has over 100k images. And was parsed under the batch size of 128, for optimality.

In this image you can see a variety ofimage quality, different variations and some images being very hard to read or review, such that, as stated above, one of the key process and factors to image generation is the idea that one has the best quality data for a network to read and to understand.

provided subsection is to evaluate the dataset that was used:

Sure, here is a reformatted version of the table:

TABLE 1 Number of dataset images for each hair and eye color tag

Hair color	**blonde**	**brown**	**black**	**blue**	**pink**	**purple**
green						
Count	4991	6659	4842	3289	2486	2972
1115 height**Hair color**	**red**	**silver**	**white**	**orange**	**aqua**	**gray**
long						
Count	2417	987	573	699	168	57
16562						
Hair style	**short**	**twintails**	**drill**	**ponytail**	**blush**	**smile**
open mouth						
Count	1403	5360	1683	8861	4926	5583
4192						
Accessories	**hat**	**ribbon**	**glasses**	**blue eyes**	**red eyes**	**brown eyes**
green eyes						
Count	1403	5360	1683	8861	4926	5583
4192						
Eye color	**purple**	**yellow**	**pink**	**aqua**	**black**	**orange**
Count	4442	1700	319	193	990	49

4.2.0.1 Label Synthesis

In section 9 the image preparation and the performance of a tag estimation through visualization, such that they are all approximation, the application of applying Illustration2Vec feature extractor was used in this instance to featch and generatee certain features. Further more, it largely shares th earchetecture and weights of Illustration2Vec tag estimator on each emage for a 4096 dimension feature vector, and the project feature vector onto a 2d space using UMAP as presented in 9. It shows the results of those images, and due to the variety of images, I believe it to be a good fit regarding training and estimation.

During the initial training of DCGANs for Label Synthesis, I encountered labeling issues while working with human and dog data. This problem indicated that there might be inconsistencies in the training set, which could potentially impact the quality of the generated images. To address this issue, I hypothesized that using a larger dataset and incorporating tags as references could improve the training process and lead to better results.

As a consequence, I decided to use the Illustration2Vec for the anime dataset. Not only is Illustration2Vec an appropriate choice for managing and organizing large datasets, but it also provides a more suitable approach for handling anime images. By utilizing Illustration2Vec and its tagging system, the training set's quality was enhanced, and the GANs' performance improved, ultimately leading to more accurate and consistent generated images.

Although this was found to be partially true, as referenced in section 3, that majority of the issues that occoured were heavily dependent on a multitude of factors. Results of using illustration2vec May be found above.

4.2.0.1.1 Reason for label synthesis label synthesis is a method for computer-generated images that generates anime faces from specific labels. This technology is helpful because it can generate realistic human faces without requiring vast datasets or intricate neural networks. This is advantageous since it makes it possible to build anime faces fast and effectively while keeping the required degree of realism. Moreover, active label synthesis may build anime faces with a specific set of attributes. This is achieved by the use of labels or tags that indicate certain facial characteristics, such as eyes, nose, mouth, and hair colour. Without individually creating each face, it is feasible to make anime faces with the appropriate appearance and feel using this approach.

Additionally, label synthesis is advantageous because of its capacity to produce anime faces from many sources. For instance, it may be used to produce anime faces from photographs already stored in a database or from im-

ages uploaded by the user. This is helpful since it allows users more control over the process of making anime faces, enabling them to precisely tailor the picture to their desired appearance.

4.3 Illustration2Vec

As previously mentioned, I2V is a deep learning-based model designed for the extraction of semantic information from images, particularly for illustrations, such as anime and manga artwork. It uses a Convolutional Neural Network (CNN) architecture, pretrained on a large dataset of tagged images to predict semantic tags and generate feature vectors that represent the content of the input images. In the context of this project, I2V was employed on the anime dataset to extract meaningful features and improve the performance of the GANs. Mathematically, the I2V model can be described as a function F_θ with parameters θ, which maps an input image I to a set of tags T and a feature vector V:

$$(T, V) = F_\theta(I)$$

The training process involves minimizing a loss function $\mathbf{s}, which measures the difference between the predicted tags and the ground truth tags, using stochastic gradient descent or a similar optimization algorithm.

Given that I2V is based on a CNN, the computational complexity of processing an image is determined by the number of layers, the number of filters (kernels) per layer, the spatial dimensions of the filters, and the size of the input image. For a 64×64 input image, the computational complexity can be represented as:

$$O\left(L \times N \times K^2 \times W \times H\right)$$

When

- L is the number of layers in the CNN,
- N is the number of filters per layer,
- K is the spatial dimension of the filters (assuming square filters),
- W and H are the width and height of the input image, respectively.

As the size of the dataset increases, the time and computational resources required for feature extraction grow linearly. This is because each image must be processed independently, and the CNN needs to perform a forward pass for every image to generate the corresponding feature vector.

When switching from the 20k dataset to the 100k dataset, the amount of data increased by a factor of 5. Consequently, the preprocessing step's time complexity also increased by the same factor, which can lead to a significant increase in the required processing time. In this case, the preprocessing step took over 8 hours, showcasing the relationship between the dataset size and the computational cost of feature extraction using I2V.

One of the core features for tag estimation and selection is through the fact that this is a CNN based tool for estimating different probabilities, a core factor to i2v is the ability to generated 512k different trypes of attributes / tags - Examples include:

- smile
- weapon
- happy
- red hair

As previosuly stated pre processing the data, was a heavy task, such that, I had to be very selective regarding the tags I chose, which was limited down to hair and eye colour. With an option to modify those tags in place as well. It was essential to consider the trade-offs between the quality of the extracted features and the computational cost when working with large datasets. And one of my core evaluations, which is some what trvial, in sync with my prior statement in section 7; a quick fire fix to fixing most issues regarding generative networks and the

type of outputts that they output, while intune with hyperparameters like batch size and learning rate, throwing the idea of adding more data to improve performance of the output, which to some degree that did work.

Quick Note : Mnist Dataset was not preprocessed and only needed to fetch certain labels or tabs, this was not a core issue.

5 Proposed Methods

Theoretical learning objective for the generator of competing methods under the optimal discriminator and classifier.

Method	Theoretical Learning Objective for the Generator
AC-GAN (Odena et al., 2017) [**ADDJk**]	$\min_G \max_{D,C} \mathscr{L}_{AC-GAN}(D,C,G)$
WAC-GAN (Shen et al., 2018) [**ADD**]	$\min_G \max D,C \mathscr{L}_{WAC-GAN}(D,C,G)$
DCGAN (Radford et al., 2016) [14]	$\min_G \max_D \mathscr{L}_{DCGAN}(D,G)$

(1)

where $\mathscr{L}_{AC-GAN}$, $\mathscr{L}_{WAC-GAN}$, and $\mathscr{L}_{DCGAN}$ are the loss functions for each respective model. The loss functions can be expressed as arrays:

AC-GAN:

$$\mathscr{L}_{AC-GAN}(D,C,G) = \begin{bmatrix} \mathbb{E}_{x\sim P_{\text{data}},y\sim P_Y}[\log D(x,y)] + \mathbb{E}_{z\sim P_{\text{noise}},y\sim P_Y}[\log(1-D(G(z,y)))] \\ +\lambda \cdot \mathrm{KL}\left(Q_{X,Y} \| P_{X,Y}\right) - \mathrm{KL}\left(Q_X \| P_X\right) + H_Q(Y \mid X) \end{bmatrix}$$

WAC-GAN:

$$\mathscr{L}_{WAC-GAN}(D,C,G) = \begin{bmatrix} \mathbb{E}_{x\sim P_{\text{data}},y\sim P_Y}[\log D(x,y)] + \mathbb{E}_{z\sim P_{\text{noise}},y\sim P_Y}[\log(1-D(G(z,y)))] \\ +\lambda \cdot \mathrm{KL}\left(Q_{X,Y} \| P_{X,Y}\right) - \mathrm{KL}\left(Q_X \| P_X\right) + \gamma \cdot \left(H_Q(Y \mid X) - \alpha\right)_+ \end{bmatrix} \quad (2)$$

DCGAN:

$$\mathscr{L}_{DCGAN}(D,G) = \begin{bmatrix} \mathbb{E}_{x\sim P_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim P_{\text{noise}}}[\log(1-D(G(z)))] \\ +\lambda \cdot \mathrm{KL}\left(Q_{X,Y} \| P_{X,Y}\right) \end{bmatrix}$$

Where P_Y is the true class distribution, D is the discriminator function, C is the classifier function, Q_X and $Q_{X,Y}$ are the generator distribution and joint distribution of input data X and output data Y, respectively. $H_Q(Y \mid X)$ is the conditional entropy of Y given X, and $(x)_+ = \max(0,x)$ is the ReLU activation function. In these equations, KL refers to the Kullback-Leibler divergence, which measures the difference between two probability distributions. γ, α, and λ are hyperparameters that control the relative importance of the different terms in the objective function.

Terms used:

- $\mathbb{E}$: This represents the expected value or the average value of a random variable. In the context of GANs, it represents the expected value of a certain loss function or a probability distribution.
- KL divergence (Kullback-Leibler divergence): This is a measure of how different two probability distributions are from each other. In the context of GANs, it is used to measure the difference between the generator distribution and the true data distribution, as well as the difference between the joint distributions of input data X and output data Y generated by the generator and the true joint distributions $P_{X,Y}$.
- λ: This is a hyperparameter that controls the relative importance of the different terms in the objective function. It is usually set through cross-validation or some other hyperparameter tuning method.
- $H_Q(Y|X)$: This is the conditional entropy of Y given X. It measures the uncertainty of the output data Y given the input data X. In the context of GANs, it is used to encourage the generator to produce diverse outputs given the same input.

- $\mathscr{L}$: This is the loss function for the GAN model, which is used to measure the difference between the generator distribution and the true data distribution. The objective of GANs is to minimize this loss function.
- $\min_G$: This indicates that we are minimizing the loss function with respect to the generator G.
- $\max_D$ and $\max_{D,C}$: This indicates that we are maximizing the loss function with respect to the discriminator D and classifier C, respectively. This is done to ensure that the discriminator and classifier are able to distinguish between the real and fake data, while the generator is able to produce realistic data that can fool the discriminator and classifier.

5.0.1 Parsing Data into a GAN

First, we preprocess the data from the chosen datasets, which in this case include the cats and dogs dataset and the human dataset, as described in the database section (refer back to Section **??**). This preprocessing step ensures that the data is in a suitable format for training the GAN.

During the training process, real samples are drawn from the data distribution, P_{data}, and noise samples are drawn from the noise distribution, P_{noise}. These samples are then used to train the discriminator and the generator following the DCGAN loss function mentioned in Section 5.0.3, 5.0.4 and **??**.

During the training process, real samples are drawn from the data distribution, P_{data}, and noise samples are drawn from the noise distribution, P_{noise}. These samples are then used to train the discriminator and the generator following the DCGAN loss function mentioned in Section 5.0.3.

5.0.2 Achieving Core Goals and Results

The primary objectives in the first term were to understand the underlying principles of GANs, explore their potential issues, and address these challenges. By working on the cats and dogs dataset and human datasets, valuable insights were gained into the functioning of GANs and their limitations.

Some of the core problems encountered in GANs included vanishing gradient and mode collapse. While these issues were later analyzed in more detail in the context of Auxiliary Classifier GANs (AC-GANs), the initial experience with DCGANs provided essential knowledge on potential solutions. To mitigate these problems, various approaches were employed, such as:

- Adding an extra layer to the network architecture: This can help improve the model's ability to capture more complex features in the data and address vanishing gradient issues.
- Cleaning up the data: Ensuring that the input data is consistent and of high quality can prevent issues related to mode collapse, where the generator produces only a limited variety of outputs.
- Other techniques: Exploring alternative network structures, loss functions, or regularization techniques can also help address challenges faced during GAN training.

By achieving these core goals and implementing the solutions mentioned above, the project successfully demonstrated the generation of high-quality images using GANs and established a strong foundation for further research and exploration in the field of generative models.

5.0.3 Deep Convolutional GANs

As mentioned above, DCGANs are an extension of traditional generative adversarial networks. In Section 5, a mathematical representation of the loss/objective function was presented. In this section, the components used, their application, and the core principles from this network model are discussed.

The DCGAN loss function can be expressed as:

$$\text{DCGAN:} \quad \mathscr{L}_{DCGAN}(D,G) = \left[\ \mathbb{E}x \sim P_{\text{data}} \left[\log D(x)\right] + \mathbb{E}z \sim P_{\text{noise}} \left[\log(1 - D(G(z)))\right] \ \right] \tag{3}$$

The DCGAN loss function consists of two main terms:

- $\mathbb{E}x \sim P\text{data}\,[\log D(x)]$: This term represents the expected value of the discriminator's output for real samples. It encourages the discriminator to classify real samples as real (output close to 1).
- $\mathbb{E}z \sim P\text{noise}\,[\log(1 - D(G(z)))]$: This term represents the expected value of the discriminator's output for generated samples. It encourages the discriminator to classify generated samples as fake (output close to 0), while the generator aims to generate samples that the discriminator will classify as real (output close to 1).

5.0.3.1 Cats and Dog Datasetss

Using the DCGAN architecture, the project explored the generation of realistic images of cats and dogs. The datasets chosen for this purpose consisted of a large number of images of cats and dogs, which were preprocessed and fed into the GAN following the process outlined in Section 5.0.1.

The GAN was trained using the loss function described in Section 5.0.3, with the aim of generating images that would closely resemble real cats and dogs. During the training process, the generator learned to create increasingly realistic images, while the discriminator's ability to distinguish between real and generated samples improved.

The results obtained from training the DCGAN on the cats and dogs datasets demonstrated the potential of GANs to generate high-quality images that closely resemble real-world data. These results provided valuable insights into the functioning of GANs, the challenges they may face during training, and potential solutions to address these issues.

5.0.3.2 Human Datasets

The project also employed the DCGAN architecture to generate realistic images of human faces using human datasets. Similar to the cats and dogs datasets, the human datasets were preprocessed and fed into the GAN as described in Section 5.0.1.

The GAN was trained using the same loss function as in Section 5.0.3, with the generator and discriminator learning to create and distinguish human faces, respectively. As the training progressed, the generator produced increasingly realistic human faces, while the discriminator honed its ability to identify real and generated samples.

The outcomes obtained from training the DCGAN on human datasets further demonstrated the capability of GANs to generate high-quality images that resemble real-world data. These results not only reinforced the understanding of GANs' functionality but also provided a deeper insight into the unique challenges and potential solutions associated with training GANs on different types of data.

5.0.3.3 Algorithm Used

1. Preprocess the data:
   (a) Load the dataset (e.g., MNIST, Cats and Dogs, or Human Faces).
   (b) Resize and normalize the images to a standard size and scale (e.g., 64×64 pixels, pixel values in the range $[-1, 1]$).
   (c) Randomly shuffle and partition the dataset into training and validation sets.
2. Initialize the generator G and discriminator D networks with appropriate architectures.
3. Set the number of training iterations, batch size, and other hyperparameters.
4. For some number of training iterations, do:
   (a) For k steps, do:
      i. Sample a minibatch of m noisy samples $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(m)}$ from the noise prior, $p_g(\mathbf{z})$.
      ii. Sample a minibatch of m real data examples $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(m)}$ from the preprocessed data distribution, $p_{data}(\mathbf{x})$.
      iii. Update the discriminator D by performing gradient ascent using the average loss according to the DCGAN loss function, for each pair $\mathbf{z}^{(i)}, \mathbf{x}^{(i)}$, where $0 < i < m$.
   (b) Sample a minibatch of m noisy samples $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(m)}$ from the noise prior, $p_g(\mathbf{z})$.
   (c) Update the generator G by performing gradient descent using the average loss according to the DCGAN loss function, for each $\mathbf{z}^{(i)}$, where $0 < i < m$.

(d) Optionally, evaluate the performance of the generator G and discriminator D on the validation set and adjust hyperparameters if necessary.

5. Save the trained generator G and discriminator D models.

Evidently, the Discriminator is updated for Kth Value steps before the Generator is modified. This procedure is performed repeatedly. Kth Value may be set to 1, although bigger numbers are often preferable [4] For optimization, any gradient-based learning rule may be utilised, both sgd and adam optimisers work well.

In the appendix section 14.1, I have provided a detailed representation of the two models that was considered during the first phase of the DCGAN, for the first term.

Summary of the DCGAN model is as follows:

5.0.3.3.1 First Model (Extra Layers) The first variant (DCGanVariantOne) has a generator and a discriminator with the following details:

Generator:

- Total Parameters: 2,794,496
- Trainable Parameters: 2,794,496
- Non-Trainable Parameters: 0

Discriminator:

- Total Parameters: 90,948
- Trainable Parameters: 90,948
- Non-Trainable Parameters: 0

5.0.3.3.2 Second Model (No Extra Layers) Generator:

- Total Parameters: 12,267,588
- Trainable Parameters: 12,267,588
- Non-Trainable Parameters: 0

Discriminator:

- Total Parameters: 91,076
- Trainable Parameters: 91,076
- Non-Trainable Parameters: 0

Results will be shown in Section 6.2.1.

5.0.4 Auxiliary Classifier GANs

Auxiliary Classifier GANs (AC-GANs) are an extension of the traditional generative adversarial networks that incorporate class label information into the training process. This allows for better control over the generated output, as the generator learns to generate samples conditioned on specific class labels. The AC-GAN loss function was presented in Section 5, and in this section, the components used, their application, and the core principles from this network model are discussed.

The AC-GAN loss function can be expressed as:

AC-GAN:

$$\mathscr{L}AC-GAN(D,C,G) = \left[\begin{array}{l} \mathbb{E}x \sim P_{\text{data}}\,, y \sim P_Y[\log D(x,y)] + \mathbb{E}z \sim P_{\text{noise}}\,, y \sim P_Y[\log(1-D(G(z,y)))] \\ +\lambda \cdot \mathscr{L}_{class}(x,c) \end{array} \right] \tag{4}$$

The core function for the generator and discriminater is the same as DC-GANs and can be found in section 5.0.3.

Both the generator network and the discriminater network are slightly modified from the original one from section 5.0.3.

- $\mathbb{E}x \sim P_{\text{data}}\,, y \sim P_Y[\log D(x,y)]$: This term represents the expected value of the discriminator's output for real samples with their corresponding class labels.
- $\mathbb{E}z \sim P$noise , $y \sim P_Y[\log(1-D(G(z,y)))]$: This term represents the expected value of the discriminator's output for generated samples with their corresponding class labels. It encourages the discriminator to classify generated samples as fake (output close to 0), while the generator aims to generate samples that the discriminator will classify as real (output close to 1) and correctly predict their class labels.
- $\lambda \cdot \mathscr{L}_{class}(x,c)$: This term represents the classification loss multiplied by the balancing hyperparameter λ. It encourages the discriminator to correctly predict class labels for both real and generated samples, improving the generator's ability to generate samples conditioned on specific class labels. This term contributes to the theoretical learning objective by ensuring that the generated samples not only resemble the real data distribution but also exhibit the desired class-specific attributes.
-

5.0.4.1 Anime Datasets

The anime dataset used in this project was divided into two phases to improve the data processing and labeling methods. In the first phase, data was fetched and processed directly from Danbooru. However, this approach was found to be less efficient and less accurate in terms of labeling. As a result, in late January, a decision was made to switch to a new approach for data processing and labeling.

5.0.4.1.1 Illustration2Vec The new approach involved using Illustration2Vec, a powerful tool for managing and organizing large datasets and handling anime images, as discussed in sections 7 and 4.2.0.1. By utilizing Illustration2Vec and its tagging system, the quality of the training set was enhanced, leading to improved performance of the GANs and more accurate and consistent generated images.

As mentioned in section 4.2.0.1, Illustration2Vec was chosen due to its ability to manage large datasets, as well as its suitability for handling anime images. The use of Illustration2Vec allowed for the generation of 4096-dimensional feature vectors and the projection of these vectors onto a 2D space using UMAP, as shown in Figure 9. This method provided a more efficient and accurate approach to labeling and training the GANs.

The switch to Illustration2Vec addressed the labeling issues that were encountered while working with human and dog data in the initial training of the DCGANs. By using Illustration2Vec and its tagging system, the quality of the training set was improved, ultimately leading to more accurate and consistent generated images. However, as noted in section 3, the majority of the issues encountered were heavily dependent on multiple factors, and the results of using Illustration2Vec may be found above.

5.0.4.2 Algorithm Used

1. Preprocess the data:
   (a) Load the dataset (e.g., MNIST, Cats and Dogs, Human Faces, or Anime dataset).
   (b) Resize and normalize the images to a standard size and scale (e.g., 64×64 pixels, pixel values in the range $[-1, 1]$).
   (c) For the Anime dataset, parse the images through Illustration2Vec to obtain class labels.
   (d) Create a dataset combining the images and their corresponding labels generated by Illustration2Vec.
   (e) Randomly shuffle and partition the dataset into training and validation sets.
2. Initialize the generator G, discriminator D, and auxiliary classifier C networks with appropriate architectures.
3. Set the number of training iterations, batch size, and other hyperparameters.
4. For some number of training iterations, do:
   (a) For k steps, do:
      i. Sample a minibatch of m noisy samples $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(m)}$ from the noise prior, $p_g(\mathbf{z})$.
      ii. Sample a minibatch of m real data examples $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(m)}$ and their corresponding class labels from the preprocessed data distribution, $p_{data}(\mathbf{x})$.
      iii. Update the discriminator D and auxiliary classifier C by performing gradient ascent using the average loss according to the AC-GAN loss function, for each pair $\mathbf{z}^{(i)}, \mathbf{x}^{(i)}$, where $0 < i < m$.
   (b) Sample a minibatch of m noisy samples $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(m)}$ from the noise prior, $p_g(\mathbf{z})$.
   (c) Update the generator G by performing gradient descent using the average loss according to the AC-GAN loss function, for each $\mathbf{z}^{(i)}$, where $0 < i < m$.
   (d) Optionally, evaluate the performance of the generator G, discriminator D, and auxiliary classifier C on the validation set and adjust hyperparameters if necessary.
5. Save the trained generator G, discriminator D, and auxiliary classifier C models.

5.0.4.2.2 Illustration2Vec Pre-Process Use Illustration2Vec to process the images and obtain class labels for each image. Store the output in a pickle file. Such that once this is done the process is to create a csv file that contains the mapping between the image filenames and their respective class labels, this is then further processed as the images May come in different form / sizes and hence using PIL, the images get normalised and reduced to the given file size, given formula is represented below:

$$normalized\_pixel\_value = \frac{original\_pixel\_value}{127.5} - 1$$

Through loading the csv file, the process is parsed through the creation of the dataset where a custom class is made to preprocesse each image.

5.0.4.3 Deeper Understanding of Training Process

Generator: The generator G takes as input a random noise vector $\mathbf{z} \in \mathbb{R}^n$ and a class label $y \in 1, 2, ..., K$, where K is the number of classes. The class label is usually encoded as a one-hot vector $\mathbf{y} \in \mathbb{R}^K$. The generator then concatenates the noise vector with the one-hot class label to form an extended input vector $\mathbf{v} = [\mathbf{z}, \mathbf{y}] \in \mathbb{R}^{n+K}$. The generator uses this extended input vector to produce a generated image $\tilde{\mathbf{x}} = G(\mathbf{v})$.
Discriminator: The discriminator D takes as input an image $\mathbf{x}$ (either a real image from the dataset or a generated image from the generator). The output of the discriminator consists of two parts: a probability of the input image

being real or fake, denoted as $D(\mathbf{x})$, and the predicted class label $\hat{y}$, represented by a probability distribution over the K classes, denoted as $C(\mathbf{x})$.
Training process: During the training process, we aim to minimize the AC-GAN loss function for both the generator and the discriminator. Recall that the AC-GAN loss function is represented in 5.0.4 and 5.

For the discriminator, we want to maximize the probability of correctly classifying real images as real and generated images as fake, while also correctly predicting the class labels. Thus, we update the discriminator using the gradients of the following objective function:

$$\mathscr{L}_D = \mathbb{E}_{x\sim P_{\text{data}},y\sim B_Y}[\log D(x)] + \mathbb{E}_{z\sim P_{\text{anis}},y,y\sim P_Y}[\log(1-D(G(z,y)))] + \lambda\cdot\mathscr{L}_{\text{class}}(x,c) \tag{5}$$

For the generator, we want to minimize the probability of the generated images being classified as fake and ensure that the generated images are associated with the correct class labels. We update the generator using the gradients of the following objective function:

$$\mathscr{L}_G = \mathbb{E}_{z\sim P_{\text{absec}},y\sim P_Y}[\log D(G(z,y))] + \lambda\cdot\mathscr{L}_{\text{class}}(x,c) \tag{6}$$

During the training process, the AC-GAN adjusts the parameters of both the generator and the discriminator to minimize, and the given parameters to reduce down to.

5.0.5 Gradient Penalty Auxiliary Classifier GANs

WAC-GAN:

$$\mathscr{L}_{WAC-GAN}(D,C,G) = \left[\begin{array}{l} \mathbb{E}x\sim P_{\text{data}}[D(x)] - \mathbb{E}z\sim P_{\text{noise}}, y\sim P_Y[D(G(z,y))] \\ +\lambda\cdot\mathscr{L}_{class}(x,c) + \alpha\cdot GP \end{array}\right] \tag{7}$$

The first term, $\mathbb{E}x\sim P\text{data}\,[D(x)]$, represents the expected value of the discriminator's output for real samples. The second term, $\mathbb{E}z\sim P\text{noise}\,, y\sim P_Y[D(G(z,y))]$, represents the expected value of the discriminator's output for generated samples. The third term, $\lambda\cdot L_{class}(x,c)$, represents the classification loss multiplied by the balancing hyperparameter λ. The fourth term, $\alpha\cdot GP$, is the gradient penalty term, where α is the hyperparameter controlling the strength of the gradient penalty.

The gradient penalty (GP) enforces a constraint on the gradients of the discriminator's loss function, leading to more stable and smooth gradients during the training process. By including the gradient penalty term in the WAC-GAN loss function, it helps to mitigate issues like vanishing gradients and mode collapse, improving the stability and performance of the model; this performance increase can be shown in **??**

5.0.5.1 Limitation and Discussions

5.0.5.2 Overview of this structure

5.1 Issues occurred during project scope

6 Experiments

6.1 Dataset and Evaluation Methods

6.2 DCGANS

6.2.1 Example of Generated Images

As referenced multiple times, that DCGANs and GANs are prone to taking huge amounts of time to train and to evaluate, such that this is no different for the mnist dataset, although It was easier than the second term due to the fact that the mnist dataset is already well founded and processed correctly, such that there is no need for heavy preprocessing, compared to how AC / WAC - Gans need / require.

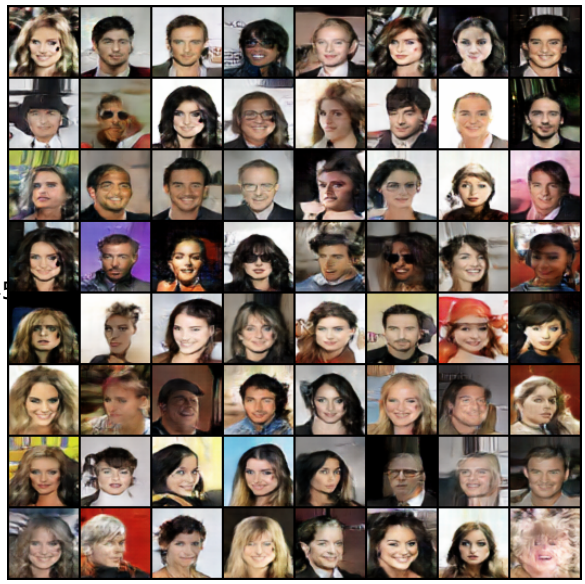
The following images were the generated outputs from my expierements in the first term. The following images were generated through 1000 epochs: Human Results of 30, 200 , 600, 1000 Epochs

30

200

media_images_Generated Images_760_f7ea902285a45

600



1k

Compared to the Cats and Dog: Results of 30, 200 , 600 Epochs

dog30.png

30

200

media_images_Generated Images_505_6b6b39b0fc21667ef079.png

600

6.2.1.0.1 Evaluation As you can see, even when manually splitting the cats and dogs dataset, with respect to the labels, clearly represented bad results. The cats and dog dataset shown above, have been split by labels, where 2 models were created, one for cats and one for dogs. It was clear from evaluating the data, that training a model with both dogs and cats presented, horrible results, even worse so that the ones listed above.

Compared to the human dataset, its clear that this provided more fruitful results, the results shown above have been trained for Humans : 1000 epochs and cats : epochs.

6.3 Implementation Details

6.4 Quantitative Results

6.5 Output Structure

6.6 Evaluation

6.6.1 Core issue of the project

7 Software Engineering

As part of developing a Generative Adversarial Network (GAN) to generate anime faces, it is important to understand the underlying principles of software engineering that enable effective development and implementation of such a project.

One of the core principles of software engineering is having a clear problem statement, which in this case is generating unique and personalized anime faces. To achieve this, extensive research was conducted by reviewing relevant papers and documenting the development process.

Some key software engineering principles and best practices that were followed for this project include:

- Data collection: Collecting and preprocessing data is a critical aspect of GAN development, as the quality and relevance of data can significantly impact the model's performance. Multiple datasets were tested, and various data preprocessing techniques were employed to identify the most viable option. Additionally, generating tags using Illustration2Vec and preprocessing images to ensure accurate tagging and cropping was a CPU-intensive process.
- Model selection: Choosing the right GAN architecture and implementing it using a deep learning framework such as TensorFlow or PyTorch is crucial to the success of the project. Upgrading the model from generating cats and dogs to humans through DCNetworks and auxiliary networks was a crucial step in improving performance.
- Hyperparameter tuning: Fine-tuning hyperparameters such as learning rate, batch size, and layer count is important for achieving optimal performance of the GAN model.
- Code organization: Given the complexity and size of machine learning projects, proper code organization is essential for effective management and maintenance. Having the right folders and scope in place is crucial for efficient development.

By adhering to these software engineering principles and best practices, the project was able to achieve its goal of generating unique and personalized anime faces using a GAN model.

7.1 ML Ops

7.2 Writing Good Code

7.3 Software Development Methodolgies

7.3.1 Waterfall model

7.3.2 Iterative Model

7.3.3 Protyping model

8 Professional Assessment

8.1 Professional Issues Intellectual property: GANs

As Generative Adversarial Networks (GANs) become increasingly capable of generating images that resemble existing works of art or photographs, intellectual property (IP) concerns have begun to surface. This emerging technology has the potential to disrupt traditional notions of copyright and IP law, as it challenges the boundaries between original and derived creations. In this essay, we will discuss some of the core problems that arise from GAN-generated images in the context of intellectual property, licensing, and other related issues.

8.2 Originality and Authorship

One of the primary challenges posed by GAN-generated images is the question of originality and authorship. According to copyright law, a work must be original and created by a human author to qualify for protection. However, GANs blur the line between human and machine authorship, raising questions about whether the generated images can be considered original works of art.

The issue of originality becomes even more complicated when GANs generate images that closely resemble existing copyrighted works. In these cases, determining the extent to which the generated image is a new, original creation or a derivative work based on the original piece becomes difficult. This ambiguity can lead to legal disputes over copyright infringement and the scope of protection for the generated images.

A specific case that highlights the complexities of IP issues in GAN-generated art is the 2018 sale of a GAN-generated painting called "Portrait of Edmond Belamy" by the French art collective Obvious. The painting sold for a staggering $432,500, far exceeding its estimated value. This case raised questions about the originality of GAN-generated works and the appropriateness of assigning copyright ownership to human creators.

8.3 Fair Use and Transformative Works

Fair use is a legal doctrine that allows for limited use of copyrighted material without obtaining permission from the rights holder. It is based on the idea that some uses of copyrighted works, such as for commentary, criticism, or parody, should be allowed to foster creativity and free expression. The concept of transformative works, which are works that add new meaning or value to the original, is central to the fair use analysis.

GAN-generated images that resemble existing works of art or photographs may be considered transformative works if they add new meaning or value to the original. However, determining whether a GAN-generated image is transformative can be a complex and subjective process. Courts may consider factors such as the purpose and character of the use, the nature of the copyrighted work, the amount and substantiality of the portion used, and the effect on the market for the original work. In the context of GAN-generated images, these factors can be difficult to assess, leading to uncertainty about whether fair use applies.

8.4 Licensing and Rights Management

GAN-generated images can also create challenges in the realm of licensing and rights management. Traditional licensing models rely on clear ownership and control of the underlying intellectual property. However, with GAN-

generated images, it can be difficult to determine who holds the rights to the generated content and who is responsible for managing those rights.

For example, consider a scenario where a GAN generates an image that closely resembles a copyrighted photograph. In this case, the rights to the generated image may be divided between the original photographer, the creator of the GAN, and potentially even the user who trained the GAN on the copyrighted image. This fragmentation of rights can make licensing and rights management complex and confusing for all parties involved.

8.5 International Implications

The challenges posed by GAN-generated images are not limited to domestic copyright law but also extend to the international arena. Different countries have different rules and standards for copyright protection, and the global nature of the internet means that GAN-generated images can easily cross international borders.

For example, some countries may recognize moral rights, which are rights that protect the personal and reputational interests of the author, in addition to economic rights. In these jurisdictions, GAN-generated images that resemble existing works of art or photographs could potentially infringe on the moral rights of the original creator, even if the generated image is considered a separate work for copyright purposes.

Furthermore, the lack of international consensus on the legal status of AI-generated works complicates matters. Some countries may be more inclined to grant copyright protection to AI-generated works, while others may adopt a more restrictive approach. As a result, creators and users of GAN-generated images may face inconsistent legal treatment depending on the jurisdiction in which their works are distributed or accessed.

8.6 Policy Considerations and Potential Solutions

As GANs continue to advance and generate increasingly realistic and complex images, policymakers and stakeholders must grapple with the intellectual property challenges that arise. Potential solutions and policy considerations include:

8.6.0.0.1 Clarifying the legal status of AI-generated works : Legislators could provide clearer guidance on the legal status of AI-generated works, including GAN-generated images, by amending copyright laws to address issues related to originality, authorship, and the scope of protection. This could involve granting limited copyright protection to AI-generated works or creating a new category of IP rights specifically for such creations

8.6.0.0.2 Alternative licensing models : Stakeholders could explore alternative licensing models to address the unique challenges posed by GAN-generated images. For example, a collective licensing approach, in which royalties are pooled and distributed among rights holders, could help simplify rights management and ensure fair compensation for creators.

8.6.0.0.3 Industry standards and best practices : The development of industry standards and best practices for the use of GANs and other AI technologies in content creation could help mitigate intellectual property concerns. Such guidelines could encourage responsible use of AI in generating images, promote transparency, and foster respect for the rights of original creators.

8.6.0.0.4 International cooperation : Policymakers and stakeholders should engage in international communication and collaboration to unify copyright rules and provide an uniform handling of GAN-generated pictures in all countries. This may include the formation of international laws or agreements that handle the special issues faced by AI-generated works.

9 Timeline

The optimal method would be to focus on core implementation during the first term and to have establish a solid foundation and working program for the second year. During the second term, emphasis will be placed on fine turning, and further tests will be conducted. Assuming all goes as planned, I will integrate more ideas to expand this project.

10 TimeLine

10.1 Term 1

10.2 Term 2

11 Proofs

12 Diary

Please be advised that this diary is not a complete record of the work done on this project, but rather a summary of the most important events and decisions made during the project. All information relating to this literature can be found on the gitlab repo.

Abbreviations

FID The Frechet Inception Distance 3

GANs Generative Adversarial Networks 3

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[2] Jaydeep T. Chauhan. *Gans vs VAE*. 2018. URL: https://www.ijcaonline.org/archives/volume182/number22/chauhan-2018-ijca-918039.pdf.

[3] Justin Engelmann and Stefan Lessmann. "Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning". In: *Expert Systems with Applications* 174 (2021), p. 114582. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.114582. URL: https://www.sciencedirect.com/science/article/pii/S0957417421000233.

[4] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[5] Minguk Kang et al. "Rebooting ACGAN: Auxiliary Classifier GANs with Stable Training". In: *CoRR* abs/2111.01118 (2021). arXiv: 2111.01118. URL: https://arxiv.org/abs/2111.01118.

[6] Jie Lei. *Animegan*. 2020. URL: https://github.com/jayleicn/animeGAN.

[7] Bing Li et al. "AniGAN: Style-Guided Generative Adversarial Networks for Unsupervised Anime Face Generation". In: *CoRR* abs/2102.12593 (2021). arXiv: 2102.12593. URL: https://arxiv.org/abs/2102.12593.

[8] Hongyu Li and Tianqi Han. *Towards Diverse Anime Face Generation: Active Label Completion and Style Feature Network*. Ed. by Paolo Cignoni and Eder Miguel. 2019. DOI: 10.2312/egs.20191016.

[9] Ziqiang Li et al. *A Comprehensive Survey on Data-Efficient GANs in Image Generation*. 2022. DOI: 10.48550/ARXIV.2204.08329. URL: https://arxiv.org/abs/2204.08329.

[10] Mattya. *Chainer-dcgan*. 2018. URL: https://github.com/mattya/chainer-DCGAN.

[11] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

[12] Augustus Odena, Christopher Olah, and Jonathon Shlens. *Conditional Image Synthesis With Auxiliary Classifier GANs*. 2016. DOI: 10.48550/ARXIV.1610.09585. URL: https://arxiv.org/abs/1610.09585.

[13] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2015. DOI: 10.48550/ARXIV.1511.06434. URL: https://arxiv.org/abs/1511.06434.

[14] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[15] Rezoolab. *Make illustration on computer with chainer*. 2015. URL: http://qiita.com/rezoolab/items/5cc96b6d31153e0c86bc.

[16] Masaki Saito and Yusuke Matsui. "Illustration2Vec: a semantic vector representation of illustrations". In: *SIGGRAPH Asia 2015 Technical Briefs* (2015).

[17] Divya Saxena and Jiannong Cao. "Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions". In: *CoRR* abs/2005.00065 (2020). arXiv: 2005.00065. URL: https://arxiv.org/abs/2005.00065.

[18] TachibanaYoshino. *AnimeGANv2*. 2020. URL: https://github.com/TachibanaYoshino/AnimeGANv2.

[19] tdrussell. *IllustrationGAN*. 2016. URL: https://github.com/tdrussell/IllustrationGAN.

Index

14 Appendix

14.1 DCGAN Layer

14.1.1 First Varient