A Comparative Study on Deep Generative Models
Generative Adversarial Networks

A forward study of anime face generation

Vivian Sedov

Interim report
BSc (Hons) in Computer Science and Artificial Intelligence
Supervised By: Dr Yunkuen Cheung

Department of Computer Science
Royal Holloway, University of London

October 2023

Contents

1 Abstract

Since the introduction of generative networks, Image generation has become ever more provident, more so, facial construction; this field has been intensively studied. Such that generative models have yielded great feats towards data distribution of training and generating new data with different variations. There have been many attempts to apply Generative Adversarial Networks (GANs) [5] and Variational Auto Encoders (VAE) [7] models to the problem of constructing new faces of anime characters, but none of the prior work have yieled positive results. In this project, I will offer a comprehensive analysis of generative models and how they vary from conventional discriminative models. The study will concentrate on the two most prevalent generative models, GANs. Consequently, I want to approach the problem of making anime faces from both a data and a model perspective by compiling a more accurate and well-suited dataset and using empirical application of unsupervised learning to apply GANs; Correctly allowing me to quantitatively analyse facial production[14]. This study will discuss the operation of several generative models, their architecture, and the experiments performed to produce new pictures. I will also elaborate on the pros and cons given by this project expenditure [4].

Keywords *Generative models, Unsupervised learning, Generative Adversarial Network, Variational Autoencoder, Machine Learning.*

2 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, great examples of this is the principle use case of up-scaling and generating images.

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[9]. 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.*

With an emphasis on parametric and nonparametric face creation techniques, both parametric *VAE* and Non-parametrices *GANs* are viable to generate face pictures, with one core difference. Parametric approaches may generate face pictures that match the form and look of a particular face. However, the created face is restricted to the form and look of the provided face, and it is challenging to generate new face shapes[16]. Non-parametric approaches, in contrast, concentrate on producing new face forms rather than adhering to the provided shape[10].

While both methods provide unique results respecting the distinct style and philosophy of the anime. Gans have seen more viable results, although gender still is a fundamental problem. Despite the scarcity of datasets, there is a vast selection of them, such as Danbooru2021 *This will be the core database that I shall be referring to in this project*, there may still be a bias of particular images over other styles. Another important factor pertaining to the dataset is its dependability and quality. Therefore, if the dataset is not accurately labelled or contains erroneous / different information, it may be hard to create a high-resolution picture and be viable in training, since GANs are notoriously tough to train. Several efforts to generate facial data from anime characters, have been documented

in existing literature. After the introduction of Geometry Consistent Generative Adversarial Network (GCGAN) by Alec Radford [13], Mattya [11] and Rezoolab [15] were the first to study the key characteristics of anime face data. Such that multiple resources came out to present these principle i.e: IllustrationGAN [17] and AnimeGAN [8], yet since the outputs from the listed have reported fuzzy and deformed images, it is still difficult to develop a high standard anime faces for anime characters.

In this project, I will attempt to bridge the gap between the fundamental needs outlined in the Introduction: Section 2 and the applicability of generative models. To overcome this fundamental problem with datasets, namely the lack of labels, I propose the notion of active label synthesis prior to feature extraction learning. This will be successful for Gans; as extracting of certain features will allow for a supervised learning process using Deep Neural Network (DNN). This will increase generalisation [9]. Thus, data gaps in the style and gender labels may be anticipated and filled with a better degree of precision than when utilising a single neural network. Although the original training data are of great quality, the quality of the produced anime faces is inconsistent. In order to satisfy the needs of practical uses, I must filter out the low-quality candidates. Still, the face picture quality is assessed using a style feature network that has been fine-tuned using face photos of varying quality. Furthermore, I want to compare the core differences between different variants of Generative adversarial networks, ranging from a basic Gan to DCGans and Conditional DCGans. To compare and to generate anime faces under the basis that the principle outlines in the Introduction:Section 2 are respected.

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

3 Scope

3.1 Aim and Scope

With my knowledge of neural networks from Machine Learning, I wanted to look into the more fundamental problems with GANs in this project. Generative adveserial networks have been growing quickly in the past few years. They are based on well-known neural networks like ML and CNN, but they offer a new way to look at and understand data. Such that I wanted to understand and compare teh generative abilities ranging from singular / Basic GANs [5] to DCGANs [cite] and many more variations.

The core problem that this project is trying to resolve is the following:

- Produced faces must be those of female and male faces, with a discernible difference between them
- Each face must have unique traits that are sufficiently distinct from the initial training batch
- The quality of the generated faces must be high[9].
- Reducing Core issues such as
  - Mode Collapse
  - Convergence issues
  - Vanishing Gradient.

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. For instance, if the customer desired an anime figure with green hair and red eyes, the final product of this project should be able to produce an image based only on those specifications.

3.2 Objectives - MileStones

3.2.1 Completed Objectives

- Understanding the mathematical principle behind GANs

- Understand the basic principle for why they are hard to train
- hyper tuning parameters and testing different network models, 2 core network models have been tested.
  - DCGANs
  - MPL GAN [Multi Perceptron Layer]
- Creating the initial anime dataset and parsing a style network on it

As of the moment, the current project has been defined as understanding, and evaluating generative adversarial networks. Furthermore, I have already loaded the initial dataset, and started working on feature extraction network required for this project to be successful.

3.2.2 Future Objectives

In this part, I will discuss the characteristics and potential next steps for my present position. This project's basic development is still in its early phases and will need more reading. While the topics I have discussed so far range from the fundamental knowledge and principle analysis of pre-data processing, it is essential to note that the global breadth of this project relies on the following fundamental principles.

3.2.2.1 Active Label Synthesis

Ideally in this study i want to seek to bridge the gap between the need for developing personalised virtual pictures and the lack of good generative models, and our contributions are twofold. such that in the future scope, it will present an active label completion strategy prior to style learning in order to overcome the problem of label shortage. Active label completion is, in essence, a weakly-supervised learning process using the aggregation of three distinct learners, deep neural networks, in order to increase generalisation capabilities. In this method, missing data elements in both the style and gender labels of gathered data may be anticipated and completed with a better degree of accuracy than if just a single neural network were used. With full labels, a style feature network (SFN) may be effectively trained to extract style characteristics for use in the next GAN.

Within the basic anime datsets that I will be using, there are three more fundamental jobs beyond the initial preset, which are as follows:

- Ground Truth label for gender, Male or Female.
- Ground Truth for style labels : Eye colour, hair colour etc [Core]
- Estimation of Image quality [Core]

To tackle the existing difficulties with picture labels, it is possible to provide a proposal of several learning approaches to actively fill the missing gender and style labels. Specifically, we manually annotate a portion of a gender label and then utilise it in conjunction with incomplete labels to train separate classifiers. In short we can use a deep neural network to verify any labels before parsing it through the generative adversearial network.

3.2.3 Pre Processing

Due to the principle that some datsets require customised images, or pre process images dependent on their categorical information:

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. Such that I will conduct experiments on Illustration2Vec [17], a cnn-based technique for predicting tags for anime illustrations. Thus, it performs a procedure very similar to the basic concept of active label Synthesis, despite the fact that the network is pretrained as opposed to a live-trained model, which is active label Synthesis.[9]

Given an anime image, the network must be able to predict the probability that it has generic anime traits, which we refer to as tags. From having a grin to, for instance, having blue hair. During the data pre-processing phase, I want to determine the optimal way to illustrate and communicate the data to the generative network and the model.

- Data pre process live, using active Synthesis
- Validate image quality
- Validate Labels
- Parse Forward to the generative adversarial network

3.3 Reason for Active label synthesis

Active 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, active 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 images 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.

Active label synthesis is an effective method for creating anime faces. It can build realistic anime faces fast and effectively from a certain set of labels.

4 Gans

4.1 What are GANs

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 lot 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[9]

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.

4.1.1 High Level Overview

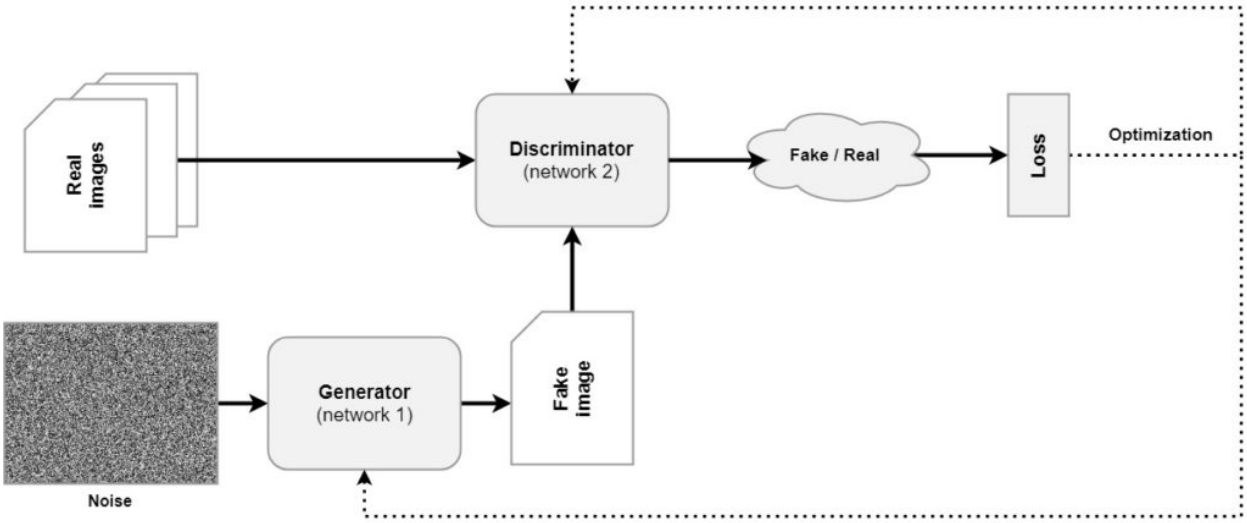


FIGURE 1 High level generative network.

Figure 1 GANs High Level Overview.

Core Loss function for Vanilla GANs

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

The discriminator D in this method attempts to maximise the output confidence score from actual samples. In addition, it reduces the output confidence score from fabricated samples provided by G. In contrast, G's objective is to maximise the D-evaluated score for its outputs, which might be seen as a deliberate attempt to deceive D.

The image it self, instates the back propagation standard where the generator receives information only about it self. Further information is listed in the next section about the core mathematical understanding for GANS.

4.2 Core Mathematical Understanding of Vanilla GAN

High level overviews are only half the understanding, and there is a deeper principle for how generative networks are presented. In this section, I will take a core deep level proof and understanding for the principles used within the project and the core mathematical understanding. To prove anything, the purpose must be explicitly (mathematically) stated. This not only guides the proof technique but also provides a target to "keep an eye out for" while reviewing the evidence. The objective is for the Generator to provide instances that are distinct from actual data. This is the mathematical idea of random variables having equal distribution (or in law). Also, their probability density functions (i.e., the probability measure produced by the random variable on its range) are identical: $p_G(x) = P_{data}(x)$. This is the precise approach of the evidence: describe an optimization issue where the ideal solution is G satisfies $p_G(x) = P_{data}(x)$.

4.2.1 Optimisation Problem

With the counterfitting example in mind, the following two words outline the reason for formulating the optimization issue. First, we must ensure that the discriminator D can identify cases from $P_{data}(x)$

$$E_{x\sim p_{data}(x)} \log(D(x)) \tag{2}$$

Where E Denotes teh expectation function SOURCE. Such that it allows us to maximise the Expectation value, to D being able to predict $D(x) = 1$ when the expectation towards x to P_{data} of x holds.

The next core term relates towards the generator G, Where the Generator is tricking the discriminator, in particular, the term comes from the negative loss loss.

$$E_{z\sim p_z(z)} \log(1 - D(G(z))). \tag{3}$$

4.2.1.0.1 Key issue One of the core issues regarding generative networks is the vanishing gradient problem, This occours when the discriminator evaluates to $-\infty$, such that if the value is maximised - where $x < 1$ is negative, meaning that $D(G(z)) \approx 0$ This is where vanishing gradient occours, and is the core principle in the algorithm listed in section LIST SECTION. The trick to avoid vanishing gradient problem, is to instead doing gradient decent on the generated data, but to do gradient accent, where you would have the inversed log function.

$$z \sim p_z(z) \log(D(G(z))). \tag{4}$$

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

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

$$E_{x\sim p_{data}(x)} \log(D(x)) + E_{z\sim p_z(z)} \log(1 - D(G(z))) \tag{5}$$

Combining the two functions listed above allows us to maximise the discriminator, with respect to G. meaning that the discriminator properly identifies real and fake data points.

Such that the optimal Discriminator will be defined as the following:

$$V(G, D) := E_{x\sim p_{data}(x)} \log(D(x)) + E_{z\sim p_z(z)} \log(1 - D(G(z))). \tag{6}$$

Where we can then further write $D^*_G = \text{argmax}_D V(G, D)$ Where the Generators aim is to reverse this. And the generator is trying to minimize the prior equation $D = D*_G$.

The original generative adverserial network paper by ian goodfellow [5] engineers this loss function into the following format :

$$V(G, D) = E_{x\sim p_{data}(x)} \log(D(x)) + E_{z\sim p_z(z)} \log(1 - D(G(z))). \tag{7}$$

Allowing an optimal solution for the Generator to be a Min Max Game on the Generator and Discriminator.

Such that at this point, we can show that the optimisation problem has a unique solution G^* and the solution must satisfies $p_G = p_{data}$.

$$E_{z\sim p_z(z)}\log(1-D(G(z))) = E_{x\sim p_G(x)}\log(1-D(x)). \tag{8}$$

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 principle will allow us to fundamentally understand how to get the best discriminator and best generator.

4.3 Best Discriminator

With the principles described above. We can write the optimal Discriminator with respect and given with the generator. Such that you would require to find a maximum of the integrated in the expanded version of the equation above. We can write the integrand as:

$$f(y) = \alpha\log y + \beta\log(1-y). \tag{9}$$

Allowing us to find the core critical points which are :

$$f'(y) = 0 \Rightarrow \frac{\alpha}{y} - \frac{\beta}{1-y} = 0 \Rightarrow y = \frac{\alpha}{\alpha+\beta} \tag{10}$$

Where if $\alpha+\beta \neq 0$ we can continue and test the second derivative shown below:

$$f''\big(\frac{\alpha}{\alpha+\beta}\big) = -\frac{\alpha}{(\frac{\alpha}{\alpha+\beta})^2} - \frac{\beta}{(1-\frac{\alpha}{\alpha+\beta})^2} < 0 \tag{11}$$

Such that we can say when $\alpha,\beta \in (0,1)$ and so is $\frac{\alpha}{\alpha+\beta}$ we can state this becomes our maximum point on the gradient *Gradient Accent* For the Discriminator. This can then be rewritten in the following manner.

$$V(G,D) = \int_x p_{data}(x)\log D(x) + p_G(x)\log(1-D(x))\,\mathrm{d}x \tag{12}$$

$$\leq \int_x \max_y p_{data}(x)\log y + p_G(x)\log(1-y)\,\mathrm{d}x. \tag{13}$$

Now allowing for $D(x) = \frac{p_{data}}{p_{data}+p_G}$, the maximum value can be achieved since the function has a unique maximiser on the interval of interest. The optimal D will also be unique with respect to the above principle.

4.3.0.0.1 Key Take Notice, that the optimal value for **D** is not practically calculable, but remains mathematically significant Since we are unaware that P_{data} is a priority, we could never utilise it directly during training. On the other hand, its presence allows us to demonstrate that an optimum G exists, which we need only estimate throughout training.

4.4 Best Generator

With that in mind, $P_G \equiv P_{data}$ Implies the following principle to be applied:

$$D_G^* = \frac{p_{data}}{p_{data}+p_G} = \frac{1}{2}. \tag{14}$$

What this means is that the discriminator is completely confused, and we have read the equilibrium point, in which, the discriminator is confused about outputing $\frac{1}{2}$ - its unsure that the data its receiving is fake or not, and ideally this is what we want to show, and get, as this is the equilibrium point. What this allows is the global minimum of training loss $\mathbb{L}(\mathbb{G}) = \max_{\mathbf{D}} V(\mathbb{G},\mathbb{D})$ to be acheived if $P_G = P_{data}$

By breaking down the original loss fucntion, we can deduce the following

$$1 - D_G^*(x) = \frac{p_G(x)}{p_G(x)+p_{data}(x)}. \tag{15}$$

This is evaluated from the following:

$$V(G,D_G^*) = \int_x p_{data}(x)\log\frac{1}{2} + p_G(x)\log\big(1-\frac{1}{2}\big)\,\mathrm{d}x \tag{16}$$

4.5 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.

4.5.1 Convergence

Convergence is a core issues, within my project, at the time of writing this, I have worked with a fully connected multi perceptron layer, neural network for both the generaator and the discriminator, and further experimentd on different network types, and fine tuning hyper parameters. Convergence is a core issues that was stated in the requirements of this project, as they are known to have a large degree of uncertainty when it comes to monitoring data. Such that with the principle understanding of Gans, i can say that the optimum of $\max_D V(G,D)$ is only valid if $P_G = P_{data}$

4.6 Current Implementation

4.6.1 Implementation / Algorithm Used

- For some number of training iterations, do:
  - For k steps, do:
    - Sample a minibatch of m noisy samples $\{\mathbf{z}^{(1)},\mathbf{z}^{(2)},...,\mathbf{z}^{(m)}\}$ from the noise prior, $p_g(\mathbf{z})$.
    - Sample a minibatch of m real data examples $\{\mathbf{x}^{(1)},\mathbf{x}^{(2)},...,\mathbf{x}^{(m)}\}$ from the data generating distribution, $p_{data}(\mathbf{x})$..
    - Update the Discriminator $D$by performing gradient ascent using the average loss according to the minimax formula above, for each pair $\{\mathbf{z}^{(i)},\mathbf{x}^{(i)}\}$, where $0 < i < m$.
    - Sample a minibatch of m noisy samples $\{\mathbf{z}^{(1)},\mathbf{z}^{(2)},...,\mathbf{z}^{(m)}\}$ from the noise prior, $p_g(\mathbf{z})$.
    - Update the Generator G by performing gradient descent using the average loss according to the expected generator loss from the formula above, for each pair $\mathbf{z}^{(i)}$, where $0 < i < m$.

[12, 5]

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 [5] For optimization, any gradient-based learning rule may be utilised, both sgd and adam optimisers work well.

4.6.1.1 Issues with this Algorithm / Modifications

Some of the core issues that have been further explained and represented within the next section is, mode collapse, unstable training and worsening image quality (Over a period of time) , Examples of this is demonstrated within the results section 4.9.

There are a few modifications on the core algorithm for DCGANs and Vanilla Gans are are quite viable :

- As stated in the mathematical portion of this project, by doing gradient asscent on the generator, instead of doing gradient decent, will allow us to diminish or reduce the overall principle of the vanishing gradient problem.
- Use label smoothing or noise to improve performance. Label smoothing helps reduce the sensitivity of the model to specific labels in the training data, while noise helps the model capture more subtle patterns.

- Use a warmup phase for your generator training. A warmup phase helps the generator learn faster and more accurately by gradually increasing the magnitude of the gradients it is receiving
- Use a variety of loss functions for the generator and discriminator. Different loss functions can help the model capture different kinds of features
- Experiment with different optimizers and learning rates. Different optimizers and learning rates can help the model learn faster or more accurately
- Use a variety of data augmentation techniques to help the model generalize better. Data augmentation can help the model learn to recognize patterns in data that it has not seen before
- Regularly evaluate the model performance to identify any potential issues. Regular evaluation of the model performance can help identify potential issues or areas of improvement

When it comes to the generative adverseral network [12] stated that to one of the core ways of avoiding mode collapse would be through having 1: a concolutional layer, due to the increased complexity, 2: Due to the unique training methods to the discriminator, its is often seen to be more stable than other generative networks. This project will be building from that. In addition to the unique training methods, DCGANs also provide rather high quality images, with respect to the image size, as seen within the results section of this report.

4.7 DCGans

While working with basic models for generative networks, I wanted to explore a further understanding, and to see if expanding a basic neural network, to convolutional networks would provide fruitful results.

Such that i decided to work with DCGANs[12](Deep Convolutional Adversarial networks)

Within this paper, a list of guidlines was stated to follow through to create a viable generative network.

- Replace all max pooling with convolutional strides
- Use transposed convolution for upsampling.
- Eliminate fully connected layers.
- Use batch normalization except for the output layer for the generator and the input layer of the discriminator.
- Use ReLU in the generator, except for the output, which uses tanh.
- Use LeakyReLU in the discriminator.

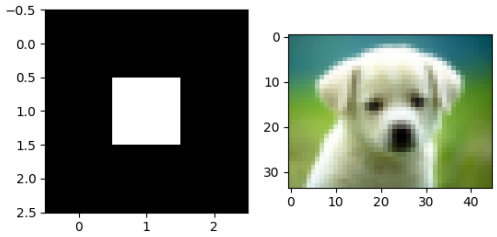
[12]

4.7.1 Batch Normalization

BatchNormalization has become known as another way to regularise, but there is no hard evidence to support this. It just seems to work better, just like RMSprop. Before sending the results of each layer to the next layer, BatchNormalization makes sure that they are all the same. This makes training go faster and seems to help people do better. As of right now, there is no agreement on whether it should go before or after the activation function, so I put it before. This theory was not only based from [5] but further reading from [1] provided similar advice regarding the batch normalisation.

4.7.2 Convolutional networks

Before beginning to comprehend how DCGans function, I want to refresh my past knowledge of convolutional networks and CNNs. And wanted to see the variations between numerous kernels and their effects on the image. This was a crucial aspect of the study as it analysed and described the optimal kernel size in relation to feature extraction.



Within this initial section of the project, I wanted to fully understand why and how convolutional networks work, more so a deep view on the code behind it. For this section, i decided to use scipy for the convolve image function

- from scipy.signal import convolve2d Convolve two 2-dimensional arrays. Convolve ‘in1‘ and ‘in2‘ with output size determined by ‘mode‘, and boundary conditions determined by ‘boundary‘ and ‘fillvalue‘.
- from skimage.color import rgb2gray : Function to compute luminance of an rgb image.
- from skimage.transform import rescale Function that performans an interpolation to upscale or downscale any N dimensional image.

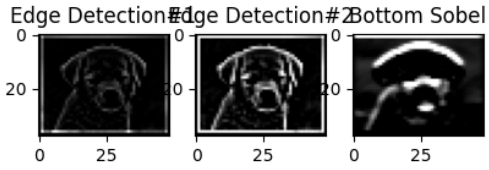
However, what are a filter and a kernel?

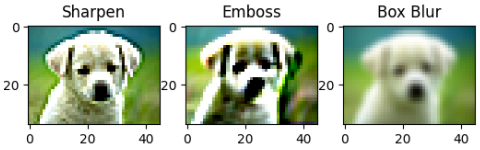
These are matrices that may be applied to a picture in order to add visual effects using the mathematical procedure known as convolution. This is the process of adding each pixel of an image to its nearby neighbours using the kernel’s weighting scheme.

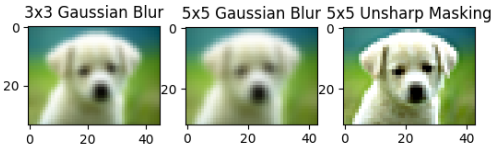
Pixel values in the output picture are determined by multiplying each kernel value by the pixel values of the input image. This procedure is repeated until the kernel has iterated this multiplication over the whole input picture.

Given the above image : i wanted to guarantee that the effects of the filters and kenels are visible, I shrunk the picture to 10 percent of its original size.

```
my_dog = imread(path)
scaled_dog = np.stack([rescale(my_dog[:, :, i], 0.10) for i in range(3)],
    axis=2)
my_dog_gray = rescale(rgb2gray(my_dog), 0.10)
\'\'\'
Here we are using the convolve2d function from scipy.signal to convolve
    the image with the kernel
the kernel is the filter that we want to apply to the image
the rescale function is used to resize the image to 10% of its original
    size
\'\'\'
def rgb_convolve2d(image, kernel):
    red = convolve2d(image[:, :, 0], kernel, "valid")
    green = convolve2d(image[:, :, 1], kernel, "valid")
    blue = convolve2d(image[:, :, 2], kernel, "valid")
    return np.stack([red, green, blue], axis=2)

identity = np.array([[0, 0, 0], [0, 1, 0], [0, 0, 0]])

conv_im1 = rgb_convolve2d(scaled_dog, identity)
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ax[0].imshow(identity, cmap="gray")
ax[1].imshow(abs(conv_im1), cmap="gray")

plt.show()
```







In this phase, I used six edge detection filters that allowed me to identify the picture's edges, as seen in the preceding left figure.

Observing the generated pictures, we can observe that edge detection only identifies regions where there is a sharp shift in hue or intensity. A high number denotes an abrupt shift, whereas a low value denotes a gradual transition. In addition, Sobel operators are analogous to edge detection, with the exception that they have a certain orientation. For instance, the bottom Sobel highlights the borders of the object's bottom portion, and vice versa. This is also presedented when we apply a blur filter, we can see the distinct outputs that it presents, and all depdentdependent on teh sharpness of the image it self.

What this means, is that data pre processing when evaluating an generative network is vital, distinguishing bad from good images, can break or make a convolutional neural network, which is further explained in teh next section.

```
kernels = [kernel1, kernel2, kernel3, kernel4, kernel5, kernel6]
```

```
kernel_name = ["Edge Detection#1", "Edge Detection#2", "Bottom Sobel", "
    Top Sobel", "Left Sobel", "Right Sobel"]
figure, axis = plt.subplots(2, 3, figsize=(12, 10))

for kernel, name, ax in zip(kernels, kernel_name, axis.flatten()):
    conv_im1 = convolve2d(my_dog_gray, kernel[::-1, ::-1]).clip(0, 1)
    ax.imshow(abs(conv_im1), cmap="gray")
    ax.set_title(name)

kernels = [kernel7, kernel8, kernel9, kernel10, kernel11, kernel12]
kernel_name = ["Sharpen", "Emboss", "Box Blur", "3x3 Gaussian Blur", "5x5
    Gaussian Blur", "5x5 Unsharp Masking"]
figure, axis = plt.subplots(2, 3, figsize=(12, 10))

for kernel, name, ax in zip(kernels, kernel_name, axis.flatten()):
    conv_im1 = rgb_convolve2d(scaled_dog, kernel[::-1, ::-1]).clip(0, 1)
    ax.imshow(abs(conv_im1), cmap="gray")
    ax.set_title(name)

plt.show()
```

4.8 Comparative view on different GANs types

Within the original paper, and the proof presented within the mathematical portion of this paper: when P_G and P_{data} can potentially have a *None overlap* support, such that the Kensen-shannon Divergence [5] in the original paper, will end up having an objective that is constantly zero, which leads to instability and issues. Further issues regarding these factors will be stated in section 4.9. [2] argued that there may exist no equilibrium in the game between generator and discriminator. One possible remedy is to use integral probability metric(IPM) based methods instead, e.g. Wasserstein distance [6], Cramer distance[3]. Some recent GAN variants suggest using gradient penalty to stabilize GAN training [3, 5, 6] Mattya [15] compared several recent GAN variants under the same network architecture and measures their performance under the same metric.

Using his work as a baseline, it stated that DRAGAN can provide more probable outcomes than other GAN variations and has the lowest computational cost. Meaning going onwards from this project, I will be further testing different generative models, with respect to this paper and its references.

Keeping this knowledge in mind, I will refer to Matty's study and build upon his foundation for enhancing a generative network beyond the boundaries of picture quality, which was my primary assumption and issue that I want to tackle.

4.9 Gan Issues

There are several issues that generative adverserial networks have: There are three core issues

1. Mode Collapse

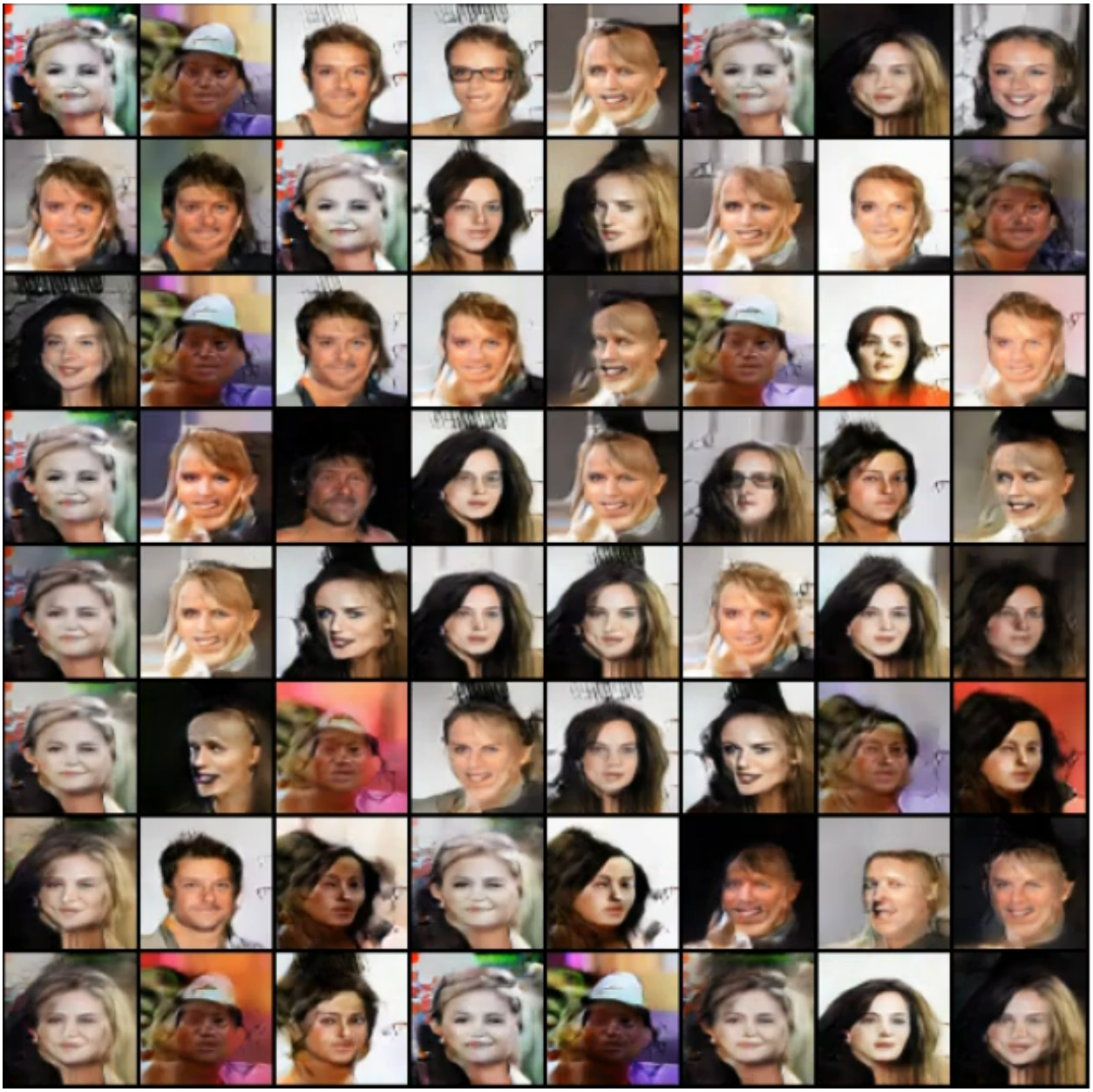


FIGURE 2 Mode Collapse : Epoch 1000.

A GAN that has been properly trained may create a range of outputs. When making photos of human faces, you would like the generator to produce batches of faces with distinct appearances and characteristics. When the generator can only create a single kind of output or a limited number of outputs, mode collapse occurs. This may occur owing to training errors, such as the generator discovering a form of data that readily fools the discriminator and continuing to generate just that type. Since there is no incentive for the generator to alter its output, the whole system will over-optimize on that one output. A great example is the figure listed above, where I had trained a network for over 3 days, and the results after 600 epochs started presenting mode collapse.

Based on my studies and observations, there is no correct method for assessing mode collapse. There was

no uniform evaluation procedure for mode collapse. Even though quantifiable metrics exist, I have been playing with them and connecting them with incoming data and usual gradient patterns.

Evaluating the score between a pre-trained model and the current model with respect to a collection of object classes is another method for preventing mode collapse. There are solutions for this, and although they cannot confirm mode collapse, they may serve as a good indicator of when it happens. This is shown by Frechet Inception Distance for a specific class of characteristics.

Examining produced photos reveals the predicted property of mode collapse, namely a large number of similar instances created independent of the input location in the latent space. It just so happens that we have drastically reduced the dimensions of the latent space to induce this outcome.

2. Convergence issues DCGANs and standard GANs have poor convergence due to mode collapse and other instability. When a GAN creates the same or similar output for all input samples, this is known as mode collapse. Techniques such as label smoothing, introducing noise to the input data, using Wasserstein distance, and mini-batch discrimination may be used to address these challenges. In addition, a number of architectural modifications, including the use of deeper networks, batch normalisation, additional convolutional layers, and more training data, may be implemented. Finally, more complex loss functions such as hinge loss, least squares loss, and perceptual loss may be used. Few of the listed fixes have been applied within the code base, but a fundemental issue that i observerd is datasets being validated with correct labels, evaluate to a functioning model. To test these theories, I ran an modular epoch system, where I could pause and retrieve data as presented with ease, and compare it to every 100th epoch. This helped with both mode collapse and convergence issues. Reaching the Nash equilibrium is the core goal in this project.

   [2]

3. Vanishing Gradient problem

   Individuals have opted to employ a separate gradient step for the generator to avoid gradients from fading when the discriminator has high confidence.

$$\Delta\theta = \nabla_\theta \mathbb{E}_{z\sim p(z)}\left[-\log D\left(g_\theta(z)\right)\right]$$

   First, we specify and demonstrate which cost function is being optimised by this gradient step. Later, we demonstrate that while this gradient does not always suffer from vanishing gradients, it does result in enormously unstable updates (which have been extensively seen in reality) in the presence of a noisy approximation of the ideal discriminator. [1]

   Let $\mathbb{P}_r$ and $\mathbb{P}_{g_\theta}$ be two continuous distributions, with densities P_r and P_{g_θ} respectively. Let $D^* = \frac{P_r}{P_{g_{\theta_0}}+P_r}$ be the optimal discriminator, fixed for a value θ_0[3]. Therefore,

$$\mathbb{E}_{z\sim p(z)}\left[-\nabla_\theta \log D^*\left(g_\theta(z)\right)\big|_{\theta=\theta_0}\right] = \nabla_\theta\left[KL\left(\mathbb{P}_{g_\theta}\|\mathbb{P}_r\right) - 2JSD\left(\mathbb{P}_{g_\theta}\|\mathbb{P}_r\right)\right]\big|_{\theta=\theta_0}$$

   [1]

   Examples of this principle being applied and the instability of the loss function can be presented with the following results.

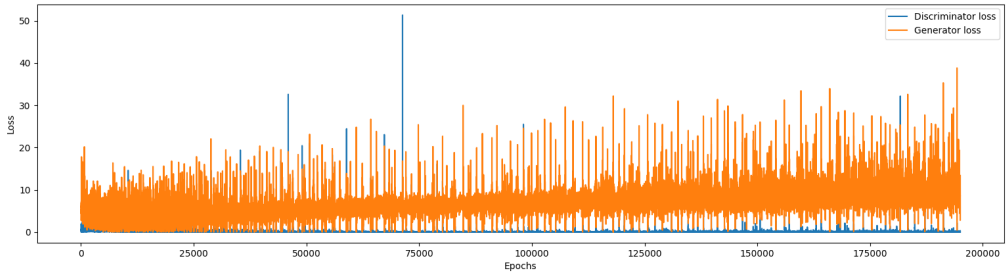


FIGURE 3 Mode Collapse : Epoch 1000.

5 Experiments

5.1 Training Details

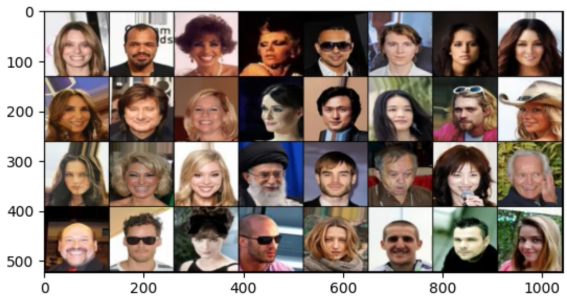
5.2 Current Model architecture

```
==========================================================================
Layer (type:depth-idx)                             Param itshape#
==========================================================================
Generator                                          --
    Sequential      : 1-1                              --
            ConvTranspose2d      : 2-1                    819,200
            BatchNorm2d      : 2-2                        1,024
            LeakyReLU      : 2-3                          --
            ConvTranspose2d      : 2-4                    2,097,152
            BatchNorm2d      : 2-5                        512
            LeakyReLU      : 2-6                          --
            ConvTranspose2d      : 2-7                    524,288
            BatchNorm2d      : 2-8                        256
            LeakyReLU      : 2-9                          --
            ConvTranspose2d      : 2-10                   131,072
            BatchNorm2d      : 2-11                       128
            LeakyReLU      : 2-12                         --
            ConvTranspose2d      : 2-13                   3,072
            Tanh      : 2-14                              --
==========================================================================
Total params: 3,576,704
Trainable params: 3,576,704
Non-trainable params: 0
==========================================================================
==========================================================================
Layer (type:depth-idx)                             Param itshape#
==========================================================================
Discriminator                                      --
    Sequential      : 1-1                              --
            Conv2d      : 2-1                           3,072
            LeakyReLU      : 2-2                        --
            Conv2d      : 2-3                           131,072
            BatchNorm2d      : 2-4                      256
            LeakyReLU      : 2-5                        --
            Conv2d      : 2-6                           524,288
            BatchNorm2d      : 2-7                      512
            LeakyReLU      : 2-8                        --
            Conv2d      : 2-9                           2,097,152
            BatchNorm2d      : 2-10                     1,024
            LeakyReLU      : 2-11                       --
            Conv2d      : 2-12                          8,192
            Sigmoid      : 2-13                         --
==========================================================================
Total params: 2,765,568
Trainable params: 2,765,568
```

```
Non-trainable params: 0
_________________________________________________________________
```

This is a fairly simple model, taken from [12, 5], the model is to some degree almost identical, as before trying on more complex datasets I wanted to comprehend and review, the simpler network, which would be the foundation for this network.

5.2.1 Model Training



At the time of writing, this model, listed in **??**. Was tested on 2 datasets. The cats and dogs dataset that was labelled but poorly proccessed with poor image quality and the human dataset; Which was correctly proccesd, but with bad labels.

The training was done on the following epochs for each dataset, with learning rates of 0.001 - 0.005 [hyperfine tuned hyperparameters from wand.ai]

- 30
- 100
- 200
- 500
- 1000

While training this model, I noticed a few things that should be properly presented.

5.2.1.1 Cats and Dogs

It is challenging to train GANs on cats and dogs datasets since they are very tiny and complicated. To learn how to produce realistic pictures, GANs need a huge quantity of data, yet the cats and dogs dataset is insufficient to meet this requirement. In addition, there are several small distinctions between cats and dogs, such as fur colour, pattern, and facial expressions, that are challenging for GANs to duplicate effectively.

Core issues that were presented :

- Imbalanced Class Distribution: The cats and dogs dataset may have an unequal representation of cats and dogs images, resulting in the DCGAN being trained on a skewed data set.
- Difficulty in Convergence: DCGANs are notoriously difficult to train, and the cats and dogs dataset may contain a large number of images which may pose a challenge to the DCGAN's ability to converge.
- Overfitting: When training a DCGAN, it is possible that the model may overfit on the cats and dogs dataset, resulting in poor performance when tested on new data.
- Computing Power: Training a DCGAN on a larger dataset such as the cats and dogs dataset may require a lot of computing power and time.
- Lack of Diversity: The cats and dogs dataset may lack diversity in the images and attributes, making it difficult for the DCGAN to generate new, interesting images.

5.2.1.2 Human

It is simpler to train human faces using generative networks on DCGANs since the networks can produce realistic-looking faces from a given picture collection. This is achievable because to the capacity of generative networks to learn facial characteristics and produce new faces. However, dealing with labels is more problematic since they are not always consistent and obvious. Labels may be subjective, making it challenging for networks to comprehend and appropriately interpret them. Moreover, labels might be imprecise and difficult to classify precisely.

Core issues relating to human faces.

- Overfitting: Overfitting occurs when the model is too closely fit to the training data, reducing its ability to generalize to unseen data. This can lead to spurious results and poor performance on the test set. To avoid overfitting, regularization techniques such as Dropout, L2 regularization, and data augmentation can be used.
- Mode Collapse: Mode collapse occurs when the generative model fails to capture the entire distribution of the data, instead focusing on a single mode. This can lead to poor results as the model fails to capture the full range of variation in the data. To avoid mode collapse, it is important to use diverse training sets and a balanced batch size.
- Poor Convergence: Poor convergence can occur when the model fails to train properly, resulting in suboptimal results. To avoid poor convergence, it is important to use appropriate hyperparameters and architecture, and to monitor the training process closely.

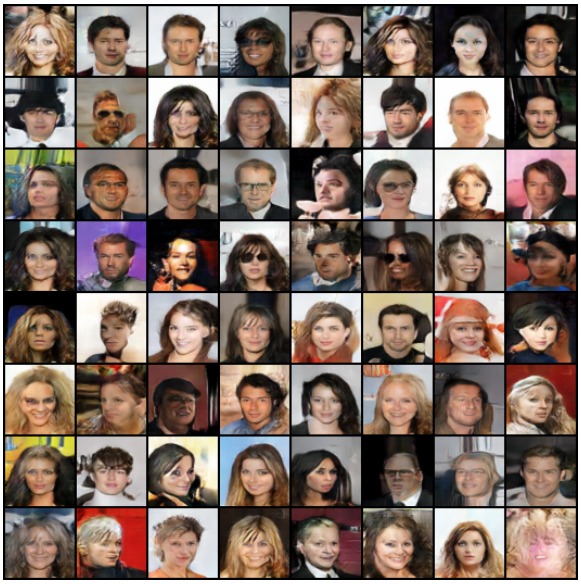
5.2.2 Generated Results

The following images were generated through 1000 epochs:

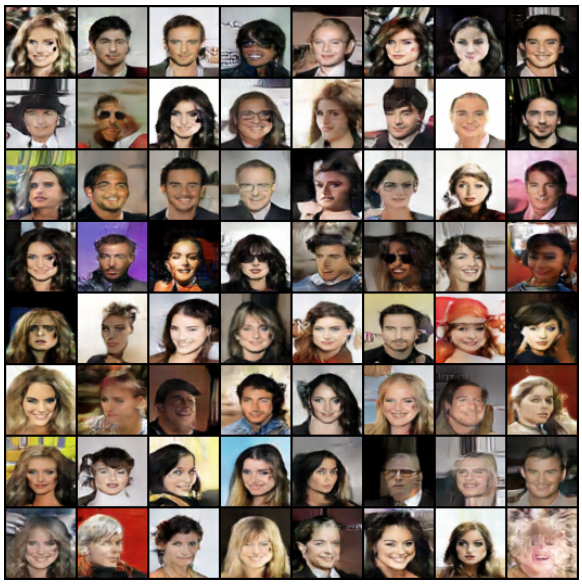
Human Results of 30, 200 , 600, 1000 Epochs

30

200

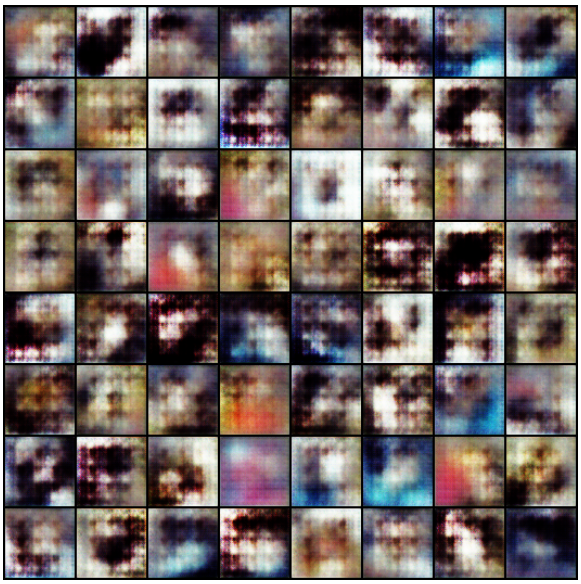


600



1k

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



30

200

600

5.2.3 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.

5.3 Quantitative Analysis

In this phase, I utilised wandb.ai to calculate gradients by calculating the cosine distance between various network weights. As indicated in the problems section, historically measuring mode collapse was quite challenging, but there are methods for evaluating such metrics, with regard to the gradient, rather than the FID distance or any variant of inception distances, compared from the original data to the output.

In the future, I will generate a wandb.ai report for each core run I do. This will function as a journal and also act as a baseline for the final report.

I will also be using further machine learning tools i.e MLOPs to evaluate the model and information. Apart from wandb.ai I will also be using *WeightWatcher.ai* Weightwatcher.ai is a tool for debugging, monitoring and analyzing the performance of Generative Adversarial Networks (GANs). It provides users with insight into the training of their GAN models. Weightwatcher.ai helps to identify problems in GAN training such as overfitting, mode collapse, and vanishing gradients. It also allows users to compare different GANs in terms of performance and also visualize their progress during the training process. Weightwatcher.ai allows users to quickly identify and address any issues that arise in the training process.

5.4 Belief / Motivation Of Model

DCGANs are essential since they are one of the most effective generative models for creating realistic pictures. They are quite straightforward to train and have a broad variety of applications, from picture production to image manipulation. DCGANs compare well to WGANS and CDCGANs, since it has been shown that they produce higher-quality pictures with more consistent training. Understanding DCGANs is crucial because they constitute the foundation for more complicated models, such as WGANS and CDCGANs. Understanding how they

function and the strategies used to train them may provide light on the operation of more complicated models. Additionally, DCGANs may be used as a baseline for assessing the performance of more complicated models.

Overall, DCGANs serve as an essential benchmark for evaluating generative networks in the job of anime face production, and their performance in this domain highlights the ability of deep learning models to generate pictures that are very realistic and diversified.

5.5 Tools

- Torch Due to its adaptability and dynamic computational graph, Pytorch outperforms TensorFlow for generative adversarial networks. It permits quicker prototyping and is easier to work with. Furthermore, Pytorch offers distributed training across numerous GPUs, making it appropriate for training on a large scale. It is better ideal for deep learning applications, such as generative adversarial networks, due to its ability to use GPUs and dynamic computational graph.

  This is particularly nice, as I am able to benefit form the ease of use, regarding multiple gpus.

- Wandb.AI wandb.ai, a software platform designed to assist academics and engineers with deep learning and machine learning difficulties, is one of the most important tools I've used for this project. It also offers a set of visualisation, logging, and detailed analysis tools to aid one's work on their model, and it gives a comprehensive perspective of their present model and the reasons why something may not be functioning as expected.

  In addition, wandb.ai's extensive capacity to link with other tools and technologies, such as pytorch, enables me to examine difficulties with my model quicker than usual.

  With my GANs, it aided me in quickly identifying any trends or patterns within the training and provides the flexibility to fine-tune hyperparameters for a specific model.

FIGURE 4 Wandb Data : Epoch 1000.

5.5.0.0.1 Examples of Wandb.Ai in action In this image, we can see the hyper parameters, and a live view of a network that was trained for over 1k epochs. Further more, it provides tools and services, to allow for hyperfine tuning, parameters, through a set of epochs, this is known as sweeps:

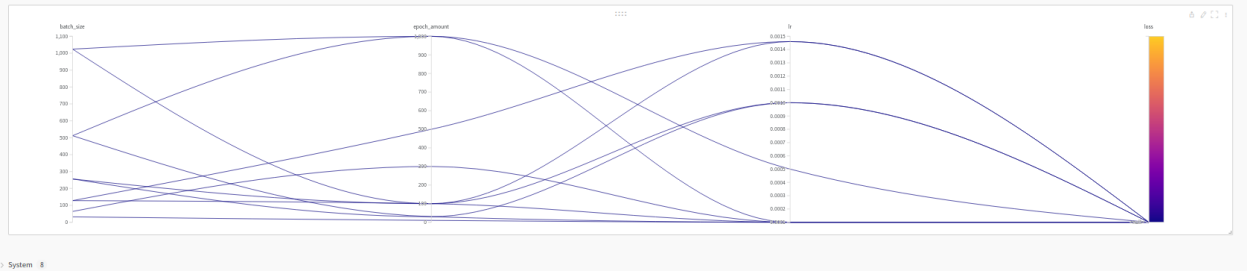


FIGURE 5 Wandb Data : Epoch 1000.

I can specify a batch size, learning rate, and epoch, and the algorithm will run over each set of possible alternatives, offering critical insight into the optimization of a functioning model. Which I found to be really handy when applying DCGAns on human faces. Although this approach has a caveat, it takes far more GPU power than I now possess.

5.6 Testing

Due to the project scope and what it is intended, test cases are very rare, or to some degree not required.

5.7 Documentation

While documentation is essential, its scope is relatively constrained based on the work I've completed, which consists mostly of analysing and comprehending the mathematical concepts behind generative adversarial networks. In light of this, I have ensured that my code has the necessary documentation and accurate comments to help me navigate through difficult code.

6 Assessment

6.1 Professional Consideration of the project

6.2 Self evaluation

7 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.

7.1 Term 1

TABLE 1 Term One

Week 1	Study more into GANs and VAE
Week 2	Formulate a full understanding of these representations

Week 3-4	Implement the code in both ML and none ML Context, and produce first Early deliverable
Week 4-5	Create a comparative view of the generated data
Week 7	Look into different variations of GANs and VAEs and compare the generated data.
Week 8-9	Fine tune and optimise my model and code
Week 10	Prepare for interim report and presentation
Week 11	Look into further detail about what I could add towards the report.

- Week 1 through week 4 Within these weeks, it was a base line for me to understand How generative networks work, more over, a high level overview, for what they are.
- Week 5 6 7 A modification against the original plan listed above, where i spent a large degree of time to review the mathematical proofs towards generative networks, instead of actively coding it . So in this instance, I was testing fully connected models and dcgans, in which i had concluded that DCGans provide better results.
- week 8 Was the week in which i started creating the model, and the code to evaluate the maths and start the data pre process on two test datasets. Cats and dogs and Human faces datasets.
- Week 9 Week 9 was taken up by allot of coursework , and had limited time to work on this project. Such that in this instance, I had decided to train and test my currentmodel, for 1000 epochs.
- week 10 and 11 Was purely based on refactoring the code, and cleaning up an impurities within the dataset using torch libs, to manipulate the dataset.

8 Diary

Please notice that the diary is published in markdown style inside the download file.

Abbreviations

DNN Deep Neural Network 3

FID The Frechet Inception Distance 2

GANs Generative Adversarial Networks 2

GCGAN Geometry Consistent Generative Adversarial Network 3

VAE Variational Auto Encoders 2

Bibliography

[1] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. DOI: 10.48550/ARXIV.1701.04862. URL: https://arxiv.org/abs/1701.04862.

[2] Sanjeev Arora et al. *Generalization and Equilibrium in Generative Adversarial Nets (GANs)*. 2017. arXiv: 1703.00573. URL: http://arxiv.org/abs/1703.00573.

[3] Marc G. Bellemare et al. *The Cramer Distance as a Solution to Biased Wasserstein Gradients*. 2017. arXiv: 1705.10743. URL: http://arxiv.org/abs/1705.10743.

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

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

[6] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028.

[7] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *CoRR* abs/1906.02691 (2019). arXiv: 1906.02691. URL: http://arxiv.org/abs/1906.02691.

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

[9] Hongyu Li and Tianqi Han. "Towards Diverse Anime Face Generation: Active Label Completion and Style Feature Network". In: *Eurographics 2019 - Short Papers*. Ed. by Paolo Cignoni and Eder Miguel. The Eurographics Association, 2019. DOI: 10.2312/egs.20191016.

[10] 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.

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

[12] 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.

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

[14] Danilo Jimenez Rezende and Fabio Viola. *Taming VAEs*. 2018. DOI: 10.48550/ARXIV.1810.00597. URL: https://arxiv.org/abs/1810.00597.

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

[16] Abhinav Sagar. *Generate High Resolution Images With Generative Variational Autoencoder*. 2020. DOI: 10.48550/ARXIV.2008.10399. URL: https://arxiv.org/abs/2008.10399.

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

Articles only

[7] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *CoRR* abs/1906.02691 (2019). arXiv: 1906.02691. URL: http://arxiv.org/abs/1906.02691.

Gans

[1] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. DOI: 10.48550/ARXIV.1701.04862. URL: https://arxiv.org/abs/1701.04862.

[2] Sanjeev Arora et al. *Generalization and Equilibrium in Generative Adversarial Nets (GANs)*. 2017. arXiv: 1703.00573. URL: http://arxiv.org/abs/1703.00573.

[3] Marc G. Bellemare et al. *The Cramer Distance as a Solution to Biased Wasserstein Gradients*. 2017. arXiv: 1705.10743. URL: http://arxiv.org/abs/1705.10743.

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

[10] 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.

[12] 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.

Misc

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

[14] Danilo Jimenez Rezende and Fabio Viola. *Taming VAEs*. 2018. DOI: 10.48550/ARXIV.1810.00597. URL: https://arxiv.org/abs/1810.00597.

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

[16] Abhinav Sagar. *Generate High Resolution Images With Generative Variational Autoencoder*. 2020. DOI: 10.48550/ARXIV.2008.10399. URL: https://arxiv.org/abs/2008.10399.

Github

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

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

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

Index