A Comparison of Different Methods of Face Image Generation

Jerry Cheng

1. Introduction

Deep generative models are a class of unsupervised machine learning models. This means that they can learn without receiving pre-assigned labels. Since they are unsupervised, these models will generally learn to approximate the probability distribution of sample data, such as music or images. Although other methods of generating images such as variational autoencoders exist, in recent years, most attention has been shifted towards generative adversarial networks (GANs) in the field of generative models. GANs are a very relevant type of neural network since they have a variety of useful applications, such as digitally aging a picture of a face or text to speech translation.

The goal of this project was to generate images of faces and to evaluate the relative performance of some of the methods used to generate faces. To complete this task, faces had to also be truly generated, rather than be copied from any learning dataset. A plain GANs, L2 GANs and WGAN were trained on the Celeba dataset, resulting in images that are recognizable as faces, but sometimes lacking in consistency. These generated images were then compared using Fréchet Inception Distance to evaluate their quality.

2. Background Information

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of neural network that utilizes a series of convolutional filters to process data (Krizhevsky et al., 2012). The convolutional filters are created by having kernels effectively scan across an input. After multiple layers, the convolutional filters are able to progressively extract higher-level information.
CNNs stand in contrast to traditional fully connected neural networks which have separate connections from every input to every output. This results in many more connections than convolutional neural networks which only need a kernel of a small size, such as 3 by 3, in order to map an input to an output. The decreased number of connections make convolutional neural networks more efficient to train and use.

Another advantage that CNNs have over fully connected neural networks are that they are able to more effectively consider spatial information. Fully connected neural networks are only able to accept 1 dimensional matrices as inputs, meaning that multidimensional inputs such as 2

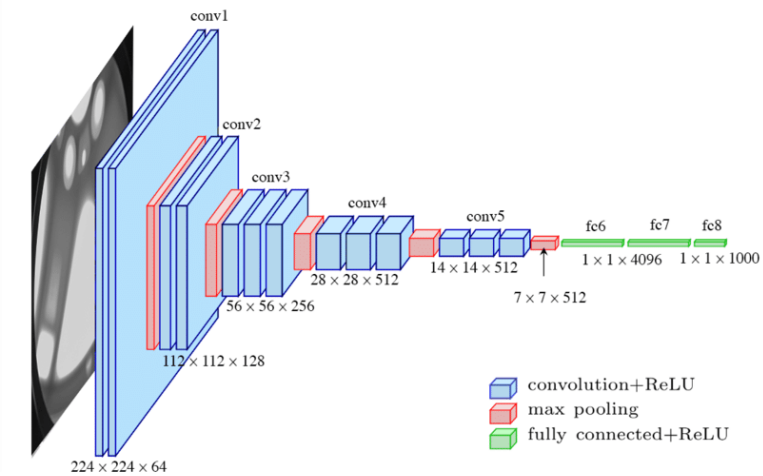

Figure 1, A typical convolutional neural network (Ferguson et al., 2017)

dimensional images must be flattened before being fed into the network. In the resulting flattened matrix, parts of the image that may have previously been next to each other may be separated, making the input more confusing for the neural network. Since CNNs can utilize kernels with multiple dimensions, they are also able to take into account information from multiple dimensions at the same time. This attribute makes them the neural network of choice for computer vision tasks or other tasks that require spatial information.

2.2 Generative Adversarial Networks

Generative Adversarial Networks consist of two separate neural networks working against each other (Goodfellow et al., 2014). The two networks are called the generator and the discriminator. The job of the generator is to take inputted random noise and generate realistic looking images, while the job of the discriminator is to look at both real images and images

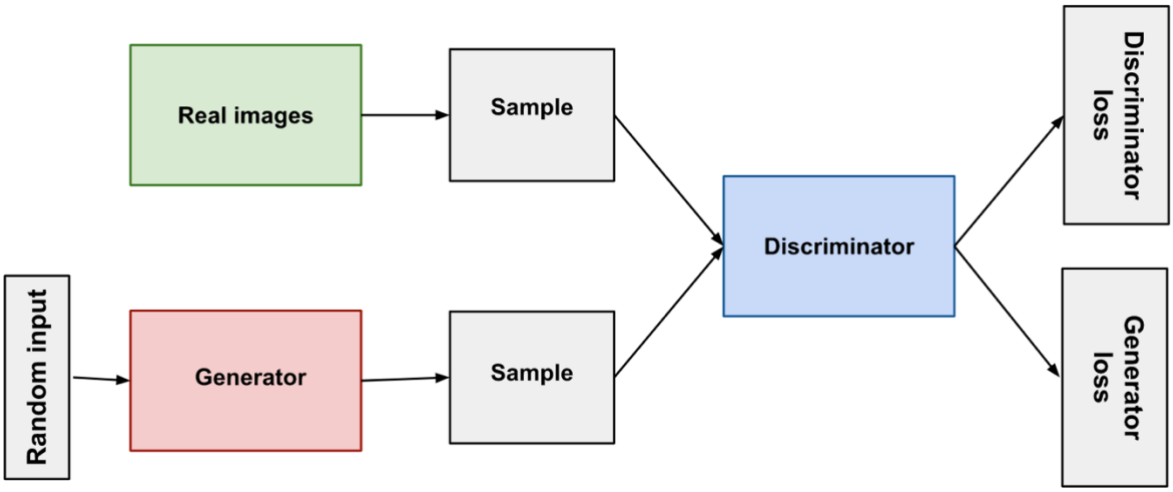

Figure 2, GAN network architecture (Overview of GAN Structure, 2019)

from the generator and correctly deduce which are real and which are fake. Figure 2 shows the structure of the whole system and the inputs and outputs of each network.

The loss function uses cross entropy and can then be calculated according to the formula below, where m is the number of images in a batch, x is the training batch of real images fed into discriminator D and z is the batch of noise that is fed into the generator G.

$$\frac{1}{m}\sum_{i=1}^{m}[\log(D(x_i)) + \log(1 - D(G(z_i)))]$$

The goal of the discriminator is to maximize the loss function, which entails maximizing both $D(x_i)$ and $1 - D(G(z_i))$. The goal of the generator is to minimize the loss function, which entails minimizing $1 - D(G(z_i))$. Note that the reason the discriminator's output is negative in the second part of the loss function is in order to match the direction at which the discriminator will attempt to optimize the loss function, as well as to make sure that the goal of the generator is opposing this direction.

In the training loop for GANs, for each batch, the generator's training will be turned off as it generates a batch of images from sampling the noise. The discriminator will then be both have fake generated images and real images from a dataset inputted and update its weights accordingly, based on whether it could distinguish real and fake images. Afterwards, the discriminator's training is turned off and the generator's training will be turned on. The generator will then sample the noise again to generate some images and update its weights accordingly, based on whether or not it could fool the discriminator. This process is repeated for all batches in the dataset and then over multiple epochs. This process is outlined as psuedocode in Figure 3.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Figure 3, GAN training loop (Goodfellow et al., 2014).

It has been mathematically proven that after training for long enough, the generator and discriminator will converge; This occurs once the generator is able to perfectly fool the discriminator. At this point, the generator will have mapped its output probability distribution to the training dataset's probability distribution, meaning that it is perfectly mimicking the dataset and thus why the discriminator is completely fooled.

GANs are commonly used for face generation due to their versatility. For example, a classifier network can be added in order to preserve identity in face generation (Shen et al., 2018) or the input can be changed to allow for perspective changes (Zou et al., 2020).

3. Proposed Improvements

GANs are notoriously difficult to train. GANs face a huge variety of problems, such as mode collapse, when training (Common Problems, 2020). Additionally, the two networks must be carefully tuned to be balanced in training so that one does not dominate the other. Many improvements to the basic GAN architecture have been proposed to remedy these issues.

3.1 Issues with GANs

One problem that occurs when training GANs is mode collapse (Monitor GAN Training Progress, 2021). This occurs when the generator maps multiple parts of the sampled noise to the same attribute, resulting in images that look very similar and do not have much variation. This allows the discriminator to very easily pick out the fake generated images from real images and the generator is not able to learn a way to fool the discriminator. This can often occur if the training data is not varied enough.

Another common problem with GANs is when one network is too strong. This results in one network dominating while the other one is unable to learn properly due to the imbalance. The common sign of this is when one network has its loss decrease very fast while the other network has its loss stay steady or increase. Usually, this problem occurs when the discriminator dominates, while the generator continually outputs images of poor quality. In this case, the discriminator's loss will quickly approach zero, while the generator's loss will either stay constant or slowly rise. This presents an issue for training, because when the loss of the discriminator is very low, this can cause gradient vanishing, which is when the gradients are too small for a neural network to optimize its weights properly.

Mode collapse can be solved by simply varying the dataset more and varying the size of the noise that the generator samples to prevent it from mapping multiple areas to the same attribute. Failure to converge due to one network being too strong can be solved by carefully tuning one network to be stronger or weaker. For example, more parameters or layers can be added to the network that is too weak or layers can be removed from the network that is too strong. Additionally, dropout layers can be added to the discriminator to weaken it. However, this tuning process can be highly time consuming and may not even be successful.

3.2 L2 Regularization

L2 regularization adds a gradient decay regularization term to the loss function. The equation is shown below (Kurach et al., 2019). The equation works by penalizing the square of the weights in the network. The square of the weight is penalized because it makes sure that the weights move towards 0 but never actually reach 0. L2 regularization also has the hyperparameter λ which controls how much the gradient decays.

$$\lambda \sum_{j=0}^{M} {W_j}^2$$

L2 regularization prevents the discriminator network from identifying high frequency noise. Thus, it should allow the generator to be trained without the discriminator identifying excessively small details of generated data and too easily labelling the images as fake. It also prevents the discriminator from picking up small details from the training images and simply memorizing them. Adding L2 regularization to the discriminator can help balance the two networks. In this sense, the L2 regularization has a similar function to dropout layers and can help balance training.

3.3 Wasserstein GANs

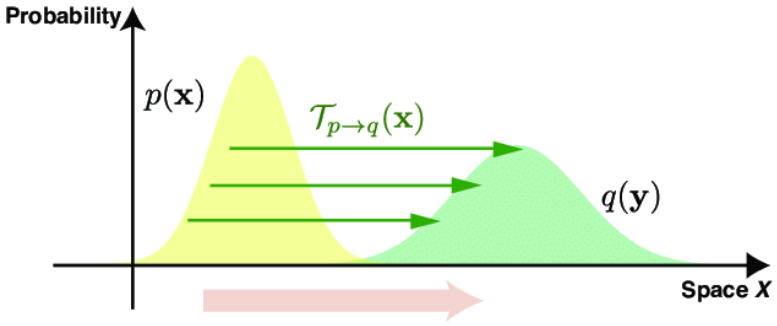

Figure 4, Visual representation of Wasserstein distance (Nakazato, M., & Ito, S., 2021)

The Wasserstein GAN (WGAN) paper proposes using Wasserstein distance (also known as earth mover's distance or EM distance) rather than the JS divergence used in traditional GANS (Arjovsky et al., 2017). EM distance can be intuitively understood as the minimum amount of "work" needed to turn one pile of "earth" into another, or the amount of minimum area of the probability distribution that must be moved multiplied by the distance which it must be moved. A visual representation of EM distance is shown in Figures 4 and 5. For example, in Figure 5, although the amount of "earth" moved is the same in both cases, the "earth" is moved over a greater distance to match h_I to $h_J{}'$ than h_I to h_J, so $h_J{}'$ has a greater earth mover distance. This loss metric is used because distributions exist that will not converge properly under JS

divergence but will converge properly under EM distance. The paper shows that under JS divergence cases exist where gradients will be 0, making it impossible for the GAN to be optimized. Another advantage of WGANs is that they make mode collapse impossible.

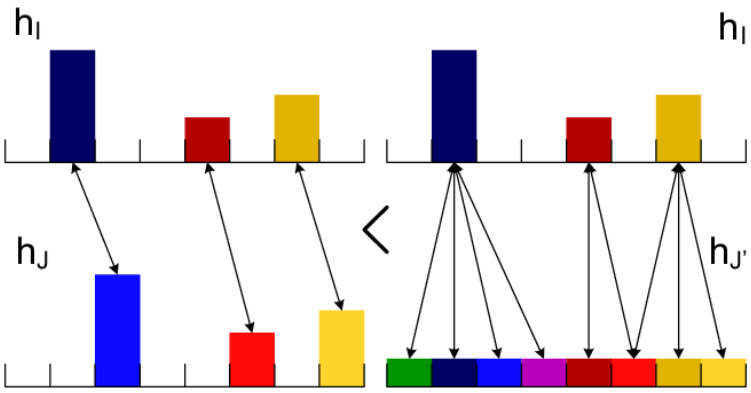

Figure 5, Another visual representation of EM distance (Assent et al., 2006)

By changing the measure of distance, a "critic" is created instead of a discriminator. The critic tries to rate images with a score of realness, rather than simply identify whether or not an image is real. In other words, the critic tries to give fake images a number that is as small as possible and it tries to give real images a number that is as big as possible.

In order to implement the critic, the last layer of the discriminator must be changed into having a linear activation function to allow it to output the larger range of scores, since activations such as the sigmoid activation would squeeze output to between 0 and 1. Additionally, the target outputs must be adjusted so that fake images have a very small target output (usually a negative number), and real images have a large target output. The discriminator should also be trained multiple times for each time the generator trains as this allows the discriminator to more quickly become an effective critic. In WGANs, it doesn't matter if the critic dominates because it will still be able to give effective feedback to the generator. In fact, it gives better feedback if it dominates. Finally, the weights of the critic must be given a constraint. This is to ensure that the critic is K-Lipchitz continuous, so that the outputs of the critic are more consistent and two similar images will have similar outputs. Although it is possible to train a WGAN without clipping the weights, this results in significantly worse performance.

4. Training the Model

The networks were trained to generate images of faces using the CelebA dataset (Liu et al., 2018). The CelebA dataset is a publicly available dataset that contains over 200,000 images of celebrities. The images are of size 178 x 218 and come pre-aligned so that the faces are centred in each image. For training, the top 20 and bottom 20 pixels of all images were cut off to form a 178 x 178 image. The 178 x 178 image was then resized into 96 x 96.

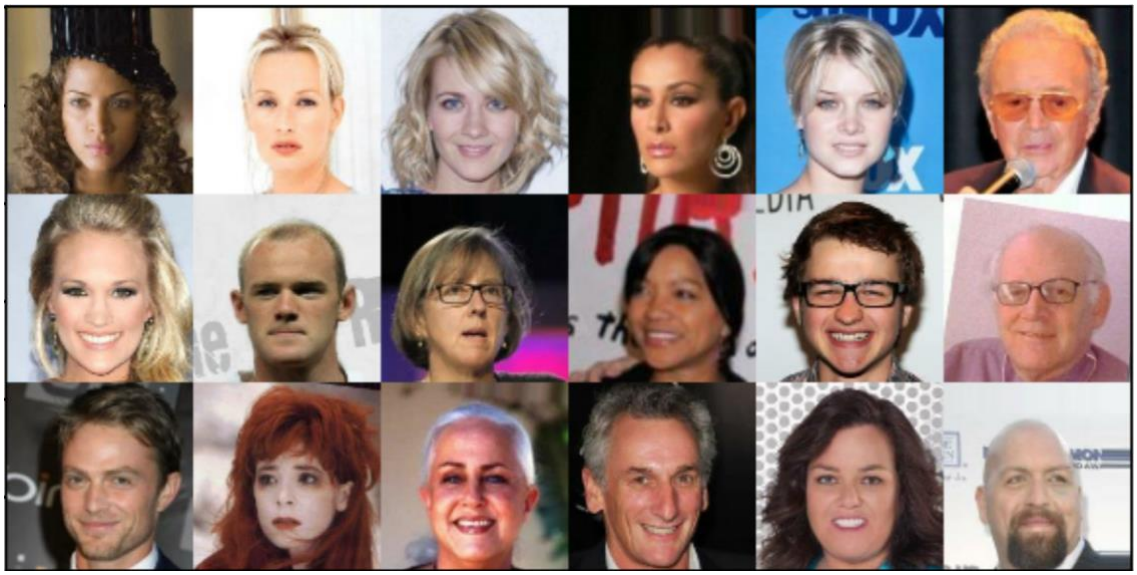

Figure 6, Examples of some images from the dataset after cropping and resizing

All the different networks used roughly the same architecture, based on the design of the plain GAN, but featured the modifications mentioned earlier. The generator outputted images with size 96 x 96, and since they were colour images, they had a depth of 3. All networks were initialized using He Normal, which initializes weights based on a normal distribution. When using Relu, normal weight initialization is very important as it prevents “dead” neurons, which are connections that do not output anything. Dead neurons occur because any negative number is turned into zero when passing through a Relu activation and thus do not convey any information.

Generator	Discriminator
$z \in \mathbb{R}^{200}$	RGB image $x \in \mathbb{R}^{96\times96\times3}$
Dense, 4x4x256	3x3 Conv2d, 32, BN, Leaky Relu
3x3 Conv2d, 256, Upsampling2d, BN, Relu	3x3 Conv2d, 64, BN, Leaky Relu
3x3 Conv2d, 256, Upsampling2d, BN, Relu	3x3 Conv2d, 128, BN, Leaky Relu
3x3 Conv2d, 256, Upsampling2d, BN, Relu	3x3 Conv2d, 256, BN, Leaky Relu
3x3 Conv2d, 128, Upsampling2d, BN, Relu	3x3 Conv2d, 512, BN, Leaky Relu
3x3 Conv2d, 128, BN, Relu	Dense 1, BN, Sigmoid
3x3 Conv2d, 3, BN, Sigmoid	

Table 1, General network design

The generator receives input of size 200 of noise. The noise is eventually scaled up into 96x96 RGB images by the generator, which are fed into the discriminator along with the cropped and rescaled images from the dataset. Additionally, for the WGAN discriminator, a linear output is used instead of a sigmoid output as the outputs don't need to be squeezed between 0 and 1.

Term	Definition
3x3 Conv2d	2D convolution with 3x3 kernel.
Upsampling2d	Filter that doubles the size of input features (images) and fills in the empty space based on neighbouring pixels.
BN	Batch normalization, a technique that normalizes inputs across a batch. It helps to accelerate learning.
Leaky Relu	Relu with a modification, where the flat section of the function is instead replaced with a section with a small gradient (e.g. 0.1) called a "leak".

Table 2, Definitions of some key terms in Table 1

Network	Optimizer
Plain GAN	Adam $lr = 0.0001, \beta_1 = 0.5, \beta_1 = 0.99$
GAN with L2 regularization	Adam $lr = 0.0001, \beta_1 = 0.5, \beta_1 = 0.99$
WGAN	RMSprop $lr = 0.0001$

Table 3, Training hyperparameters

During the training process, networks were allowed to train for 50 epochs, but training ended when the networks either converged, or it became obvious that the networks would fail to converge, which was usually at around 30 epochs. RMSprop was used for the WGAN because

the original authors highlighted problems when training with momentum-based optimizers such as Adam (Arjovsky et al., 2017).

5. Analysis of Results

There are many ways to evaluate the quality of generated images. One way is to directly evaluate images generated from random noise, which was done to get the images in this section. Another more objective method is to evaluate the distance between the generated images and the training images, which was done by using the Fréchet Inception Distance.

5.1 Plain GAN

This GAN uses the architecture proposed in the original GAN paper with no modifications. The design can be seen above. Below in Figure 7 are some random examples of images that have been generated by the plain GAN. These results can be used as a baseline to compare the other results with. Although all the images are recognizable as faces, the quality varies a lot. Some of them carry distortion and artifacting while others look almost realistic.

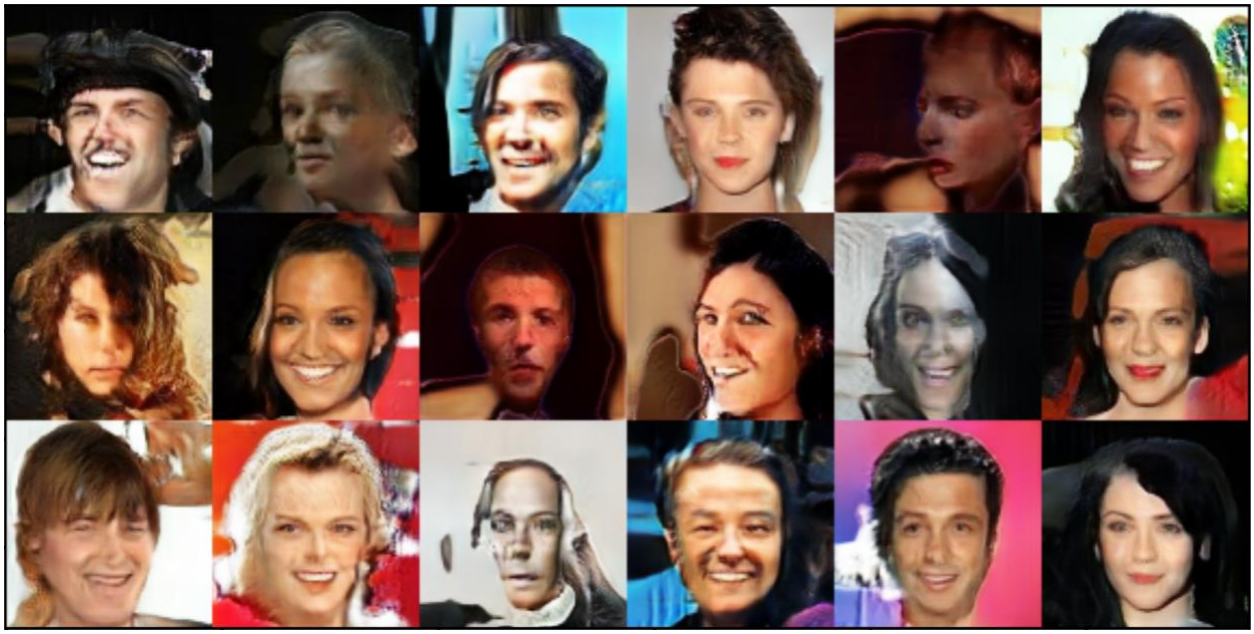

Figure 7, Sample of randomly generated images from Plain GAN

5.2 GAN with L2 Regularization

The GAN with L2 regularization, which uses $\lambda = 0.02$, appears to produce images of similar or slightly better quality to the plain GAN (more sample images can be found in the appendix). The images seem to be more consistent, with a lower number of unrealistic faces generated, but out of the good faces, there is more graininess than compared to the Plain GAN. Training with L2 regularization was also more stable, with the discriminator less likely to dominate and have its loss approach 0.

Figure 8, Sample of randomly generated images from L2 GAN

5.3 WGAN

Figure 9, Sample of randomly generated images from WGAN

Although the WGAN is theoretically able to produce better results than the plain GAN or GAN with L2 regularization, the WGAN produced far worse images than the other networks in practice. The faces are very incoherent, and some images contain two faces blended into each other. This poor quality was due to the difficulty of tuning the weight clipping constraint. The authors recommended a clip constraint of 0.01, but this value was found to be too high. With a clip constraint that is too large, such as 0.01, the loss will explode to numbers in the millions, while when the clip constraint is too small this causes poor discriminator performance because it cannot learn properly with such a strong constraint. The final network which generated the images shown was trained with a clip constraint of 0.01 and was never able to converge properly due to the huge loss.

5.4 Fréchet Inception Distance

The Fréchet Inception Distance (FID) is a more objective method of measuring the performance of GANs (Heusel et al., 2017). FID calculates the distance between two Gaussian distributions: the training images for the GAN and the images the GAN generates. The distance is calculated based off feeding the images into a classification CNN such as Inception v3, which was used in this case, and then taking the outputs of one of the deeper layers. This is done because in the deeper layers of a classification model, higher level attributes such as shapes, colour or texture should have been extracted. FID is able to pick up on aspects of images such as noise or blurring (Brownlee, 2019).

Network	Mean FID (lower is better)	Standard Deviation
Plain GAN	85.29	2.57
GAN with L2	83.29	3.17
WGAN	159.66	2.36

Table 4, comparison of FIDs for the networks tested

6. Future Work

As mentioned earlier, although WGANs should theoretically be better than normal GANs using cross entropy, the results from this investigation were unsatisfactory, mostly due to the difficulty of tuning the weight clipping hyperparameter. Additionally, the images generated were only 96 x 96, while far higher resolution images are possible with the use of other architectures.

6.1 WGAN Gradient Penalty

By the authors' own admission, clipping weights is not a very good way to enforce the Lipchitz constraint. WGAN with gradient penalty (WGAN-GP) is a type of WGAN that uses a gradient penalty to the loss function rather than a hard constraint on weights to make the function k-Lipschitz continuous (Gulrajani et al., 2017). The gradient penalty is calculated off an interpolation of real and generated images. Mathematically, all points of interpolation should be sampled, but in order to save on processing power a randomly chosen point is sampled

instead. Using the interpolation, how well the model conforms to the Lipchitz constraint can be calculated. If the model is shown to be moving away from being Lipchitz continuous, it is penalized. Additionally, the discriminator also cannot use batch normalization, as this produces correlations between different images in a batch. The authors recommend layer normalization to be used instead.

For WGAN-GP, the performance of the network is less dependent on the hyperparameter used to change the strength of the gradient penalty, thus making it easier to train the GAN and removing the difficulty of tuning the value at which weights are clipped

6.2 Progressively Growing GAN

Progressively growing GAN (ProGAN) is a method of training GANs that improves network stability, especially for higher resolutions (Karras, 2017). The network is first trained at a smaller scale and resolution and then slowly scaled up to the desired resolution, as shown in Figure 10.

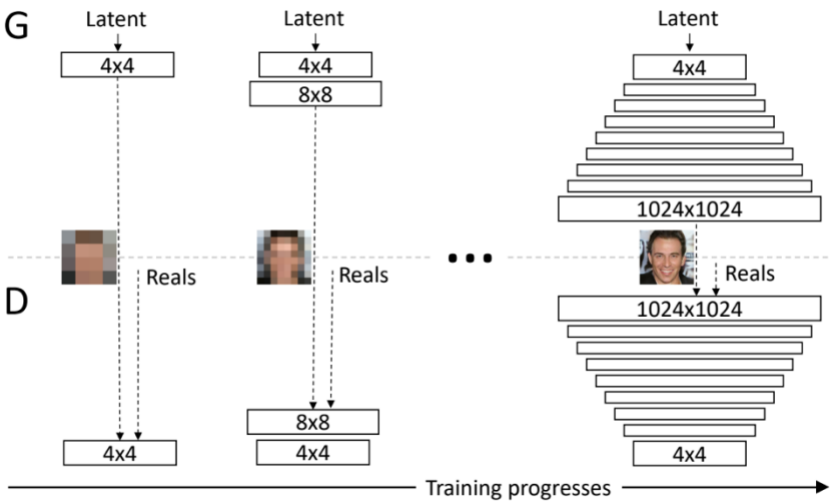

Figure 10, How ProGANs are trained

The network is grown to a higher resolution by first entirely using the previously trained network and then slowly shifting to utilizing the newly trained network, until the entire network relies on the new section. For example, in the middle of training, the output for the deepest layers of the discriminator would be half determined by the older network and half determined by the newer network.

This network could be used in the future to generate higher resolution images, or simply to improve stability when training for 96 x 96 images.

7. Conclusion

In conclusion, the goal of this project was to generate faces and evaluate the relative performance of different methods of generating faces. A plain GAN, a L2 GAN and a WGAN

were trained on the Celeba dataset, which consists of over 200,000 images of celebrity faces. The results of each network were evaluated using both subjective and objective measures such as the Fréchet Inception Distance.

A plain deep convolutional GAN was shown to already be decent at generating images and appeared to become slightly better with the addition of weight decay. A WGAN was also tested but achieved worse results than the other approaches due to the difficulty of tuning and training it.

In the future, WGAN-GP could be used, as it will mean that WGANs will be able to converge with less tuning and the loss won't grow to be huge. A ProGAN could also be tested to see if it improves network stability at 96 x 96 or higher resolutions.

8. References

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.

Assent, I., Wenning, A., & Seidl, T. (2006). Approximation techniques for indexing the earth mover's distance in multimedia databases. In 22nd International Conference on Data Engineering (ICDE'06) (pp. 11-11). IEEE.

Brownlee, J., (2019). How to Implement the Frechet Inception Distance (FID) for Evaluating GANs. Machine Learning Mastery. https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/

Common Problems | Generative Adversarial Networks | Google Developers. (2020). Google Developers. https://developers.google.com/machine-learning/gan/problems

Ferguson, M., Ak, R., Lee, Y. T. T., & Law, K. H. (2017, December). Automatic localization of casting defects with convolutional neural networks. In 2017 IEEE international conference on big data (big data) (pp. 1726-1735). IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 30.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028.

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

Kurach, K., Lučić, M., Zhai, X., Michalski, M., & Gelly, S. (2019). A large-scale study on regularization and normalization in GANs. In International Conference on Machine Learning (pp. 3581-3590). PMLR.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2018). Large-scale celebfaces attributes (celeba) dataset.

Monitor GAN Training Progress and Identify Common Failure modes. (2021). Mathworks.com. https://www.mathworks.com/help/deeplearning/ug/monitor-gan-training-progress-and-identify-common-failure-modes.html

Nakazato, M., & Ito, S. (2021). Geometrical aspects of entropy production in stochastic thermodynamics based on Wasserstein distance. arXiv preprint arXiv:2103.00503.

Overview of GAN Structure. (2019). Google Developers. https://developers.google.com/machine-learning/gan/gan_structure

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Shen, Y., Luo, P., Yan, J., Wang, X., & Tang, X. (2018). Faceid-gan: Learning a symmetry three-player gan for identity-preserving face synthesis. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 821-830).

Zou, H., Ak, K. E., & Kassim, A. A. (2020). Edge-gan: Edge conditioned multi-view face image generation. In 2020 IEEE International Conference on Image Processing (ICIP) (pp. 2401-2405). IEEE.

9. Appendix

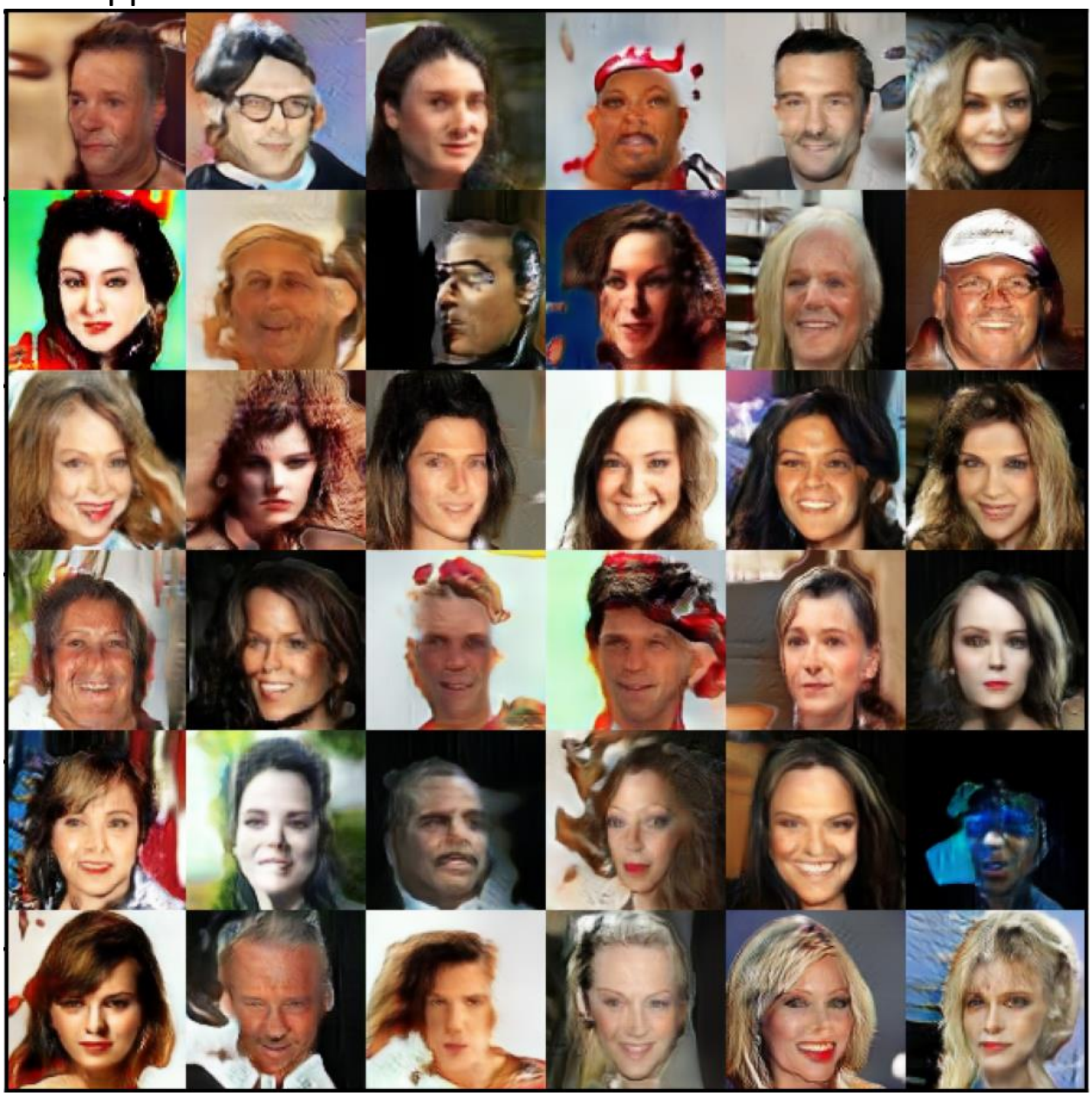

Figure 11, Other Plain GAN sample images

Figure 12, Other L2 GAN sample images

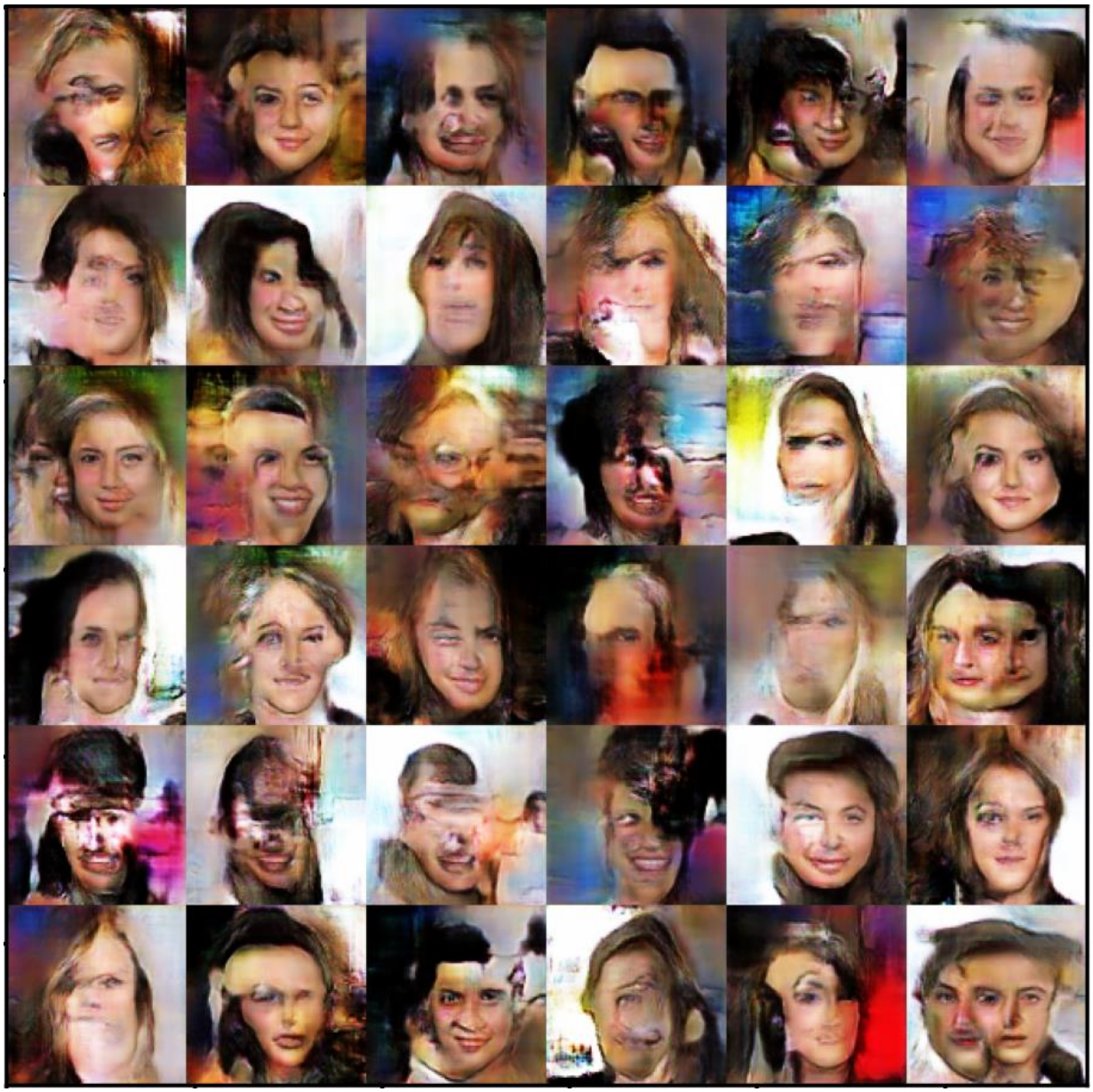

Figure 13, Other WGAN sample images