
Creating Human-like Faces using Generative Adversarial Network (GAN)

Raj Shekhor Roy

University of Missouri-Columbia
Department of Computer Science
rsr3gt@mail.missouri.edu

Farhan Quadir

University of Missouri-Columbia
Department of Computer Science
fqg7h@mail.missouri.edu

Abstract

GAN is one of the hottest topics in the field of Neural Nets if not the most. It is relatively new in comparison as well as exciting and the potential to it is endless. In this project we trained the vanilla GAN and one of its variations, the Deep Convolutional GAN (DCGAN), to produce human-like faces using the CelebA dataset. We observed that although the vanilla GAN could produce facial features to an extent, the images were very noisy and unclear in some cases. On the other hand, DCGAN managed to produce human faces with much clearer facial features than the vanilla GAN. The images were still a bit blurry, noisy, and distorted in some instances.

1 Introduction

The idea of generative models has been a subject of great interest to researchers. Many approaches to develop generative models have been coined like variational encoder (VAE) [8] or autoregressive models like PixelRNN [5], and GAN is one of the most recent additions to that group in the year of 2014 by Ian Goodfellow[1]. Since then there has been no shortage of interest in this field because of its robustness. Although all of the models have their pros and cons but among them the performance of GAN is noteworthy when it comes to generation of media data. We are often intrigued by the level of perfection achieved by GANs when we see the deepFake videos that are made with personalities like Barack Obama and Mark Zuckerberg.

The main purpose of generative models is to generate data. The use of GAN is versatile when it comes to the sector of generative data. The basic (vanilla) Generative Adversarial Nets (GANs) consist of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations within a dataset[1]. There are many variations of GAN like Super-resolution GAN (SRGAN) [6] which can be used to increase resolution of low resolution images by four times, CycleGAN [7] can be used to transform some properties of one image to another target image. These properties may not belong to the target image at all. For example, transforming the skin of a Zebra to a horse.

There are other variations like text-to-image synthesis but, for our project, we explored the vanilla GAN which is based on [1] to generate images of the faces of people. In addition to that we also explored the deep convolutional neural network GAN (DCGAN) [2] which is a heavier and powerful version of the vanilla GAN that replaces the multilayer perceptron (MLP) architecture of the vanilla GAN with deep convolutional neural networks (CNN) [4]. We used the CelebA dataset [3] to train our models. We expect that DCGAN will generate better looking images than the

vanilla GAN. Amongst the numerous uses of GANs, creating fake actors for games, anime, and advertisements are notable. Recently, GAN-inspired video compression and transmission across low bandwidth for video conferencing has been a remarkable breakthrough.

2 Background on GAN

2.1 Mechanism of GAN

GANs consist of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations and features within a dataset [1,2,6,7]. This idea of two competing models trying to beat each other allows the models to grow in skill and become better at the game. Hence, the models are termed adversaries of each other.

2.1.1 Generator Mechanism

The **generator** produces fake data $G(Z)$ from random noise Z and feeds them into the discriminator, $D(G(Z))$. If the value of $D(G(z))$ is 0 i.e the discriminator detects the fake data then its loss penalizes the generator. Then uses backpropagation to update the weights in the generator network. It does not affect any parameter of the discriminator. The learning in the generator occurs without any supervision and solely depends on the loss passed back from the discriminator.

2.2.1 Discriminator Mechanism

The discriminator is trained using supervised learning. The discriminator's training data comes from two sources: *real* data and *fake* data. The *real* data, X , are the real or true pictures of the faces of people directly sampled from the CelebA dataset. The discriminator uses these instances as positive labels (value = 1). During training, these images are fed into the discriminator to get an output $D(G(X))$. If the value of the output is 0 then it penalizes itself for misclassifying the target as the label used for true data is 1. On the other hand *fake* data ($G(Z)$) is generated from the noise (Z) by the generator. The fake images are also fed into the discriminator as negative labels (value = 0) and are penalized if the value of the output, $D(G(Z))$ is 1. The generator network's parameters are not updated during training of discriminator, and similarly, the discriminator parameters remain the same when we are training the generator. Weights are updated through backpropagation right from the discriminator into the generator by back propagating the discriminators output of the fake images to the generator loss.

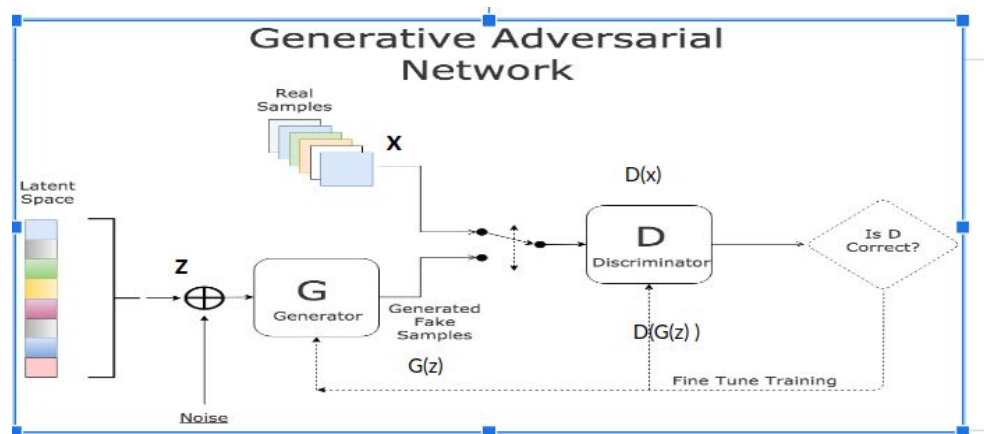


Figure 1: Illustrating the principle of the vanilla (basic) GAN

3 Methods

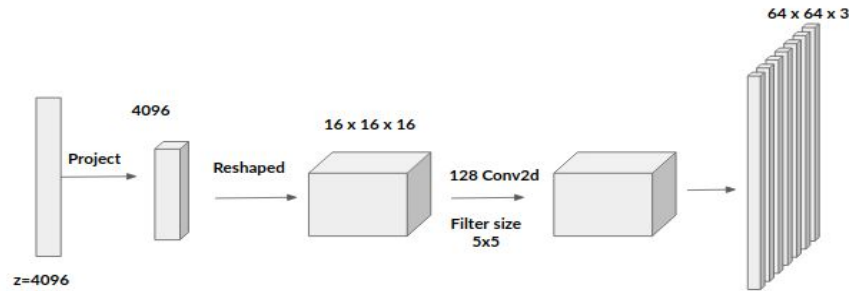
3.1 Dataset

The *CelebA* dataset which is a large-scale face attributes dataset with more than 200,000 celebrity images, each with 40 attribute annotations. The data is available from MMLAB, The Chinese University of Hong Kong (<http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>). The images in this dataset cover large pose variations and background clutter. CelebA has huge arrays, large quantities, and rich annotations, including 10,177 number of unique personnel, 202,599 number of face images, five landmark locations, and 40 binary attributes annotations per image, which was processed to crop the only human face, and hence the celebA_aligned dataset was produced which was used in this project. This dataset was leveraged as the dataset on which the original paper used ‘The Toronto Face Database’ was off limit for us and another point-worthy thing is that the faces the paper used is that the images were black and white and also the resolution of the images were only 32x32 where as ours is colored with resolution 172x218x3 .

3.2 Vanilla GAN

The vanilla GAN was designed to capture the very basic essence of the GAN. Form the Figure 2 (a) we can see that it uses the input/noise of shape of 4096 as input and then reshapes it into 16x16x16 tensor. This is fed into a 2D convolution of only 128 layers with filter size 5x5 keeping the padding same which in turn outputs the final image of size 64x64x3. On the last layer tanh activation is used and it was optimized using gradient descent. And on figure2 (b) we presented the architecture of the discriminator which uses a humble dense layer of only 512 neuron and then feed it into another dense layer of 256 neuron which are then in turn feed into the dense layer of 1 activated by sigmoid function, which is the probability of whether an image is faker or not. The discriminator is optimized using gradient ascend and on the last layer sigmoid activation is used.

(a) Generator Architecture of GAN



(b) Discriminator Architecture of GAN

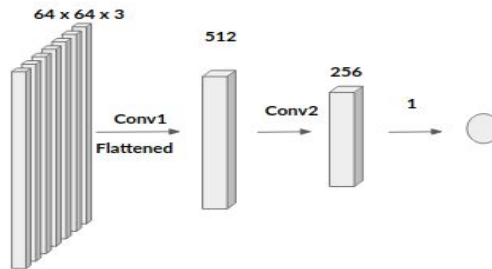
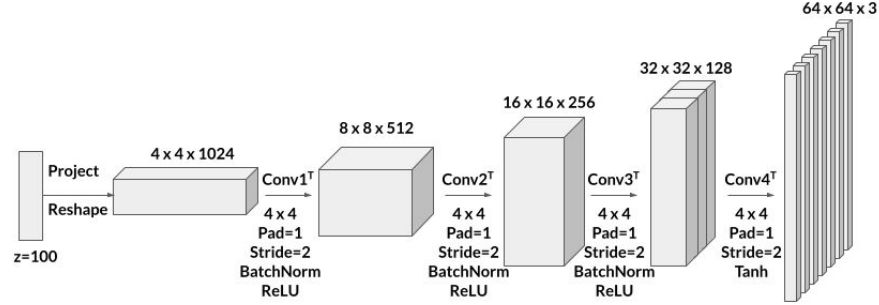


Figure 2: Illustration of the architecture of vanilla GAN (a) Generator Architecture (b) Discriminator Architecture

3.3 DCGAN

Deep Convolutional GAN (DCGAN) is built on a similar idea of the vanilla GAN. The generator and the discriminator portions of the GAN are replaced with deep convolutional neural networks (CNN) [3] instead of the multilayer perceptron of the GAN [2]. The proposed CNN architectures of the generator and the discriminator are illustrated in figure 3.

(a) Generator Architecture of DC GAN



(b) Discriminator Architecture of DC GAN

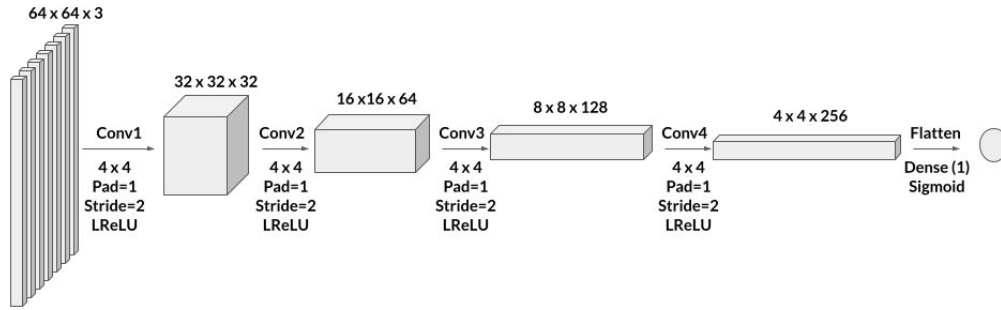


Figure 3: The DCGAN CNN network architecture used in our development. (a) The generator CNN architecture and, (b) the discriminator CNN architecture.

From figure 3(a), we can see the generator CNN architecture of the DCGAN. The input is a random noise vector of size 100. This is projected and reshaped into a tensor of dimensions $4 \times 4 \times 1024$. This tensor is then upsampled using a Convolution Transpose layer using $512 \times 4 \times 4$ filter, padding = 1, and stride = 2 followed by batch normalization. The activation function is a rectified linear unit (ReLU). This is repeated four times halving the number of filters each time, while for the fourth layer the number of filters is set to three. This upsamples and increases dimensions of the image while reducing the channels until we reach our final image size of $64 \times 64 \times 3$. This

output of the fake image is then passed to the discriminator to train using the supervision of some real images of similar dimensions $64 \times 64 \times 3$.

The discriminator takes the input of real images and fake images generated by the generator along with its labels. The discriminator (figure 3(b)) consists of four convolution layers with filter size 4×4 , padding =1, stride = 2 with leaky ReLU (LReLU) activation. The number of filters start with 32 in the first layer and is halved until the fourth layer. The last layer is flattened and fully connected to only one output neuron with sigmoid activation.

For both generator and discriminator, we used minmax logits adversarial loss function. When the discriminator trains, the discriminator's fake image loss is passed back to the generator to optimize and train. Summation of both the real image loss and the fake image loss is the total loss of the discriminator. We train for 50 epochs with Adam optimizer and learning rate fixed at 0.0002 for both generator and discriminator. The batch size was fixed at 256. The development for DCGAN was done in Python 3.6 with PyTorch 1.0. The training took approximately four days. The results are reported in section 4.2.

4 Result

4.1 Vanilla Gan

In figure 4 it can be observed that the difference of loss of the discriminator with real images and fake images keeps on decreasing with each epoch as the generator learns to mimic the real images more effectively. According to the global optimality criterion, the Gan converges when it will reach Nash-equilibrium in this min-max game i.e for all inputs the, the output of the discriminator will produce 0.5 i.e no action will cause it change its decision. But one of the biggest challenges of training a Gan is that the discriminator often becomes much stronger than the generator, which also happened in this case and the generator loss kept on increasing with each epoch. To cope with it many techniques are used in the field like training the discriminator with double learning rate or training the generator twice.

In addition to that we also produced a tile of generated images by the generator on fig 5 . The images are placed in the ascending order of epoch from left to right and the gradual change in the quality of image is visible. At the very beginning, image1 to 3 it was just noise and then slowly seemed to learn the basic outline of the face which look more like a colored shadow from image 4-7, as it progressed the colored shadow became more clear from image 8-12, from image 12 -17 the generator starts learning the details of a human face which consists of eyes,nose,mouth and hairs. From image 18 and onwards all of them look like human faces and in addition to capturing face details, soon the noise starts to decrease and by 'human understanding' we can easily classify them as a proper human-face.

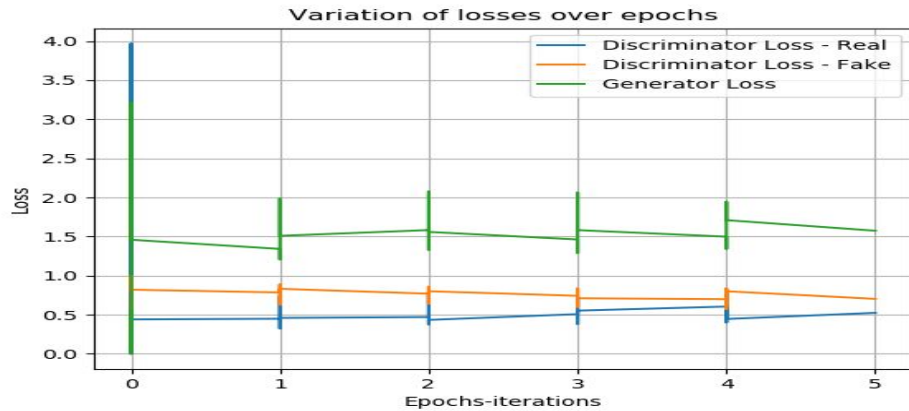


Figure 4: Illustrating the loss of the discriminator loss with real images using blue lines, the discriminator loss with fake images using orange lines and the generator loss with green lines for the first 5 epochs of the training.


























				
<i>image 1</i>	image 2	image 3	image 4	image 5
				
image 6	image 7	image 8	image 9	image 10
				
image 11	image 12	image 13	image 14	image 15
				
image 16	image 17	image 18	image 19	image 20
				
image 21	image 22	image 23	image 24	image 25

Figure 5 : Displaying image generated by the generator with increasing epoch from left to right

4.1 DCGAN

Figure 6 shows the loss for training the DCGAN while figure 7 shows 16 images generated by the generator that passed the discriminators test.

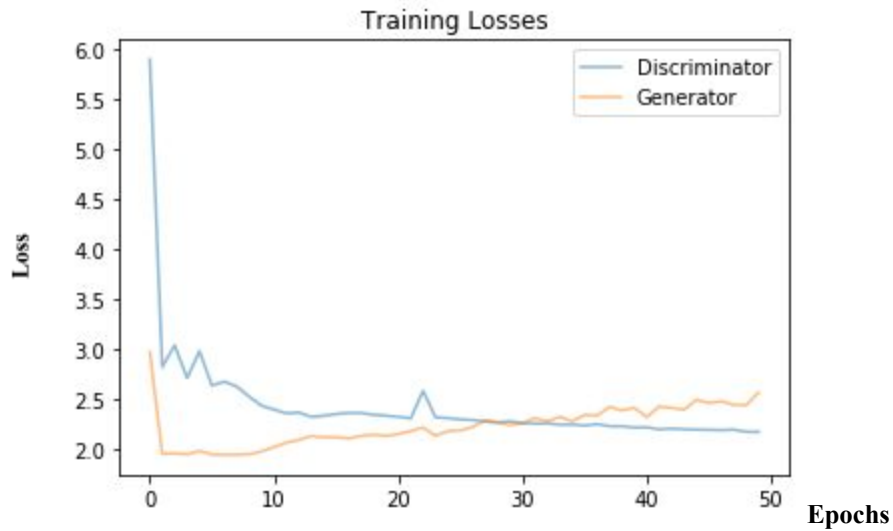


Figure 6 : Figure showing the changes in discriminator and generator losses over 50 epochs of training. We can see the generator loss increases while the discriminator loss decreases. After around 30 epochs generator loss overtakes discriminator loss.

From figure 6, we can see that as the discriminator trains, it starts with a very high loss. The loss decreases drastically after a few epochs and then steadily decreases with slight fluctuations. On the other hand, the generator also learns quickly as its loss increases steadily as well with fluctuations. After around 30 epochs, the generator loss becomes greater than the discriminator loss suggesting network convergence.

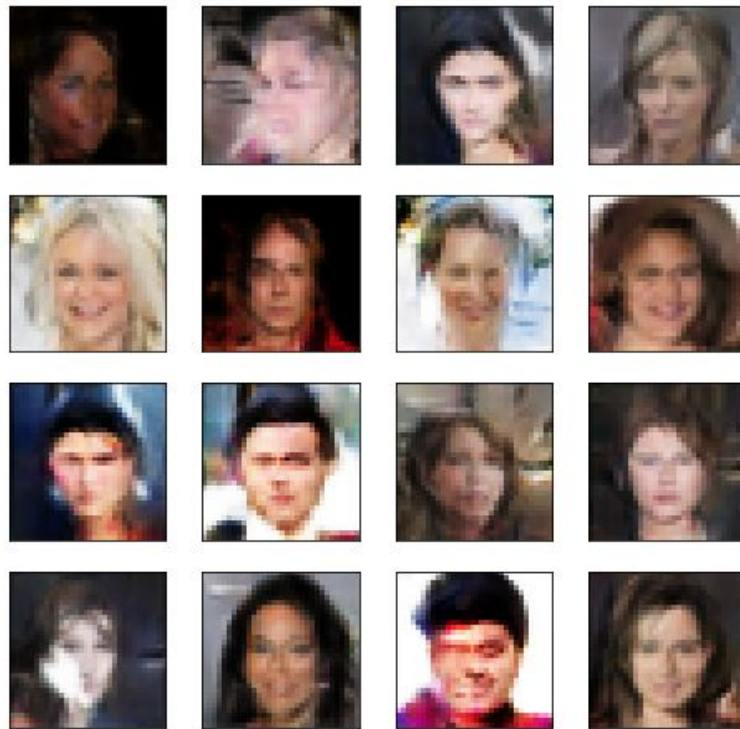


Figure 7: Sixteen images generated by the generator of the DCGAN that were passed as real images by discriminator. We can clearly see the formation of facial features and emotions. Images are still blurry.

From figure 7 we can see that the images generated by DCGAN contain much less noise than those generated by the vanilla GAN. Although the images appear blurry, the facial features of the DCGAN images are more distinct and facial expressions are also visible. In some cases we observe image distortions and merges which imply the need of a denoiser or image smoother. Also resolution improvement can be applied. Another interesting observation for both vanilla and DCGAN is that both models generate images of more females as compared to males. This may be due to the fact that the training dataset is unbalanced and contains more images of females than of males. So the generator is biased towards generating more female images. The same may be true for the discriminator, which is also biased into discriminating the male generated images as fakes.

5 Conclusion and Future Work

We successfully trained the vanilla GAN and the DCGAN models on the CelebA dataset and compared the results. DCGAN outperformed the vanilla GAN by leaps and bounds. Although generated images were blurry, DCGAN clearly produced better quality images with lesser noise and clearly visible facial features like eyes, noses, mouths, hair, etc with plenty of variation. DCGAN also managed to capture emotions like smiles and incorporate it into some of the generated images. In some cases, the DCGAN generated images had distortions and shadows. Both models generated more female images than that of males.

In the future, we would like to explore deeper layers of generators and discriminators and if possible a residual architecture. Tuning the hyperparameters and experimenting with other adversarial losses (which was not possible in this short time) could have led to better performance. Also, other types of GANs like CycleGAN [7], similarity constraint GAN (SCGAN) [9], etc. and their variations can be experimented with. Training using the high-resolution data and other datasets can also improve the performance of the networks. We also believe stacking a trained denoising VAE or denoising GAN over the generated images will lead to better quality image generation.

References

- [1] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014) Generative Adversarial Nets. *Advances in Neural Information Processing Systems* 27, pp. 2672-2680.
- [2] Radford, A., Metz, L. & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434t: Under review as a conference paper at ICLR 2016*
- [3] Liu, Z., Luo, P., PWang, X., & Tang, X. (2015). Deep Learning Face Attributes in the Wild. *Proceedings of International Conference on Computer Vision (IICV)*, pp. 3730-3738.
- [4] LeCun, Y., & Bengio, Y. (1995) Convolutional networks for images, speech and time series. *The handbook of brain theory and neural networks*. (3361), pp. 1995
- [5] van den Oord, A., Kalchbrenner, N., & Kovukcuoglu, K. (2016) Pixel Recurrent Neural Networks. *arXiv preprint arXiv:1601.06759*
- [6] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681-4690
- [7] Zhu, J.Y., Park, T., Isola, P., & Efros, A.A. (2017) Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2223-2232.
- [8] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.
- [9] X. Li, L. Chen, L. Wang, P. Wu and W. Tong, "SCGAN: Disentangled Representation Learning by Adding Similarity Constraint on Generative Adversarial Nets," in *IEEE Access*, vol. 7, pp. 147928-147938, 2019, doi: 10.1109/ACCESS.2018.2872695.