Machine Learning Project 2

Group 8

Liong Khai Jiet

1 Introduction

Deep learning-based generative models have grown in popularity in recent years as the field has advanced substantially. With huge datasets and introduction to deep neural networks, these generative networks can generate realistic material such as images and sounds. This project will focus on the two main families of the generative models, Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) to generate human faces.

1.1 Dataset

The CelebA dataset [5] is a large scale dataset of celebrity images. In this project, we would like to use this dataset explore the GANs to understand how the network works. Although there is the High Quality version of the dataset (CelebA-HQ), this dataset is used due to the limited access to GPU resource.

Besides, the project also used the `Align&Cropped Images` instead of `In-The-Wild Images` which are not preprocessed and cropped to the face. We will use the total of 202599 images for this project.

1.2 Methodology

This project is testing with different type of GANs to compare the ability of the GANs to synthesis a face image. In this report, the preprocessing methods, algorithm used and the results of the experiment are compiled according to the network are as follows: In section 2, we use another image generation technique, likelihood-based generative model, Variational Autoencoder (VAE) for further experiment and comparison. In section 3.1, the DCGan algorithm is applied as the baseline for the project.

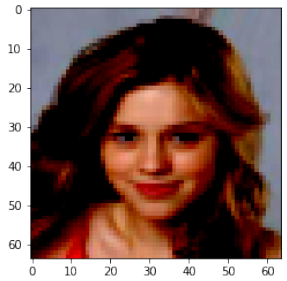
1.3 Preprocessing

The images in the dataset are with the shape of (218, 178, 3). All the data are fed into a custom `pytorch.Dataset` to ease the transformation of the images with the help of the `pytorch.vision`, which are:

- `transforms.Resize()`. This method helps to resize the image to a standard size to be feed into the network.
- `transforms.CenterCrop()`. This method helps to crop the image to the center of the image to remove possible noise such as background objects or faces as shown in figure 1a.
- `transforms.ToTensor()`. This method helps to convert the image data into Pytorch tensors for computation.

(a) Before (noisy image)

(b) Before



(c) After

Figure 1: Dataset Preprocessing

- `transforms.Normalize()`. This method is used to normalize the image data to have 0.5 mean and 0.5 standard deviation for a faster training process.

1.4 Experiment

The experiments are run using PyTorch Lightning [2] framework for the data loading and training loop implementation. The code are run using a hosted Jupyter Lab in Google Colab instance with limited free tier GPU resource.

2 Variational Autoencoder

VAE is an autoencoder whose encoding distribution is regularised during training to ensure that its latent space has good qualities that allow us to generate fresh data. The name "variational" is derived from the variational inference method in statistics.

VAE consists of two part, the encoder and decoder. The role of the encoder is to do a dimension reduction of the high dimensional image to lower dimension called latent variable. The role of the decoder is to reconstruct the input image from the latent vector as similar as possible. The model is said to be converged when the encoder keep maximum information when encoding and, decoder has the minimum reconstruction error when decoding.

For the VAE model, we implemented two ResNet18 model downloaded from Pytorch model as the encoder and decoder. The learning rate is fixed for all experiments but the dimension of latent vector is fine-tuned as a hyperparameter to tune the model with optimal reconstruction loss.

The Resnet18 model input dimension is set to the 64 x 64 x 3 dimension and the output dimension of 512 as recommended by the author [4], followed by the latent vector with dimension 256 and 512 for comparison, the decoder network follows a similar setup as the encoder but with inverse order. The latent variable here serves as a bottleneck to force the encoder to save only useful features and the decoder to reconstruct the features as close as possible. The Figure 2 shows the architecture used for this experiment. The VAE model is trained with 10 epochs.

The implementation of reconstruction loss term is the Mean Square Error Loss as the first loss term. Besides, VAE also includes KL divergence term to prevent the network from memorizing the input samples as a regularization term. The objective function of the VAE can be shown as

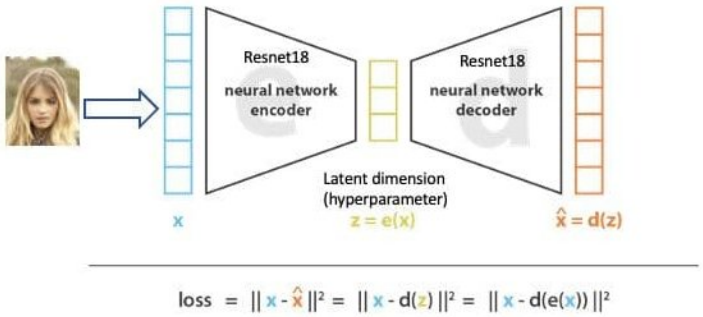
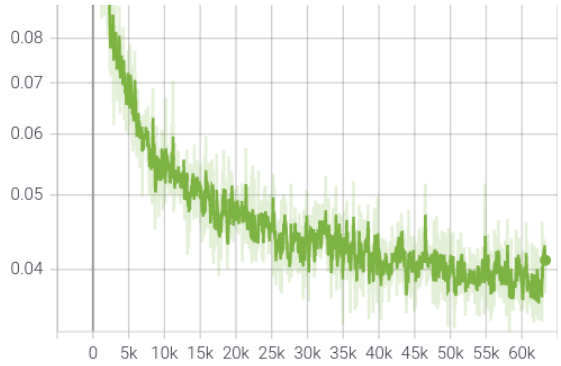


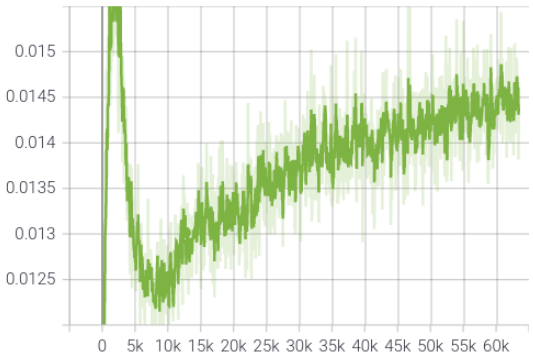
Figure 2: Overview architecture of VAE. Modified from source [1]

$$\text{Loss} = \text{loss}_{\text{recon}} + \text{Loss}_{\text{KL}}$$

and the graph is plotted in Figure 3.



(a) Reconstruction loss plot of VAE training



(b) KL Divergence between $p(z)$ and $q(z)$

Figure 3: Training loss of VAE

2.1 Attempt #1

As comparison, we also tried to train with image size of 128 to see whether the network is able to output a clearer image. However, increasing the image size multiplied the training time (about 120 minutes for one epoch). So, we train the model for only **1** epoch and choose generated image of the base model trained with 1 epoch for fair comparison.

The vae model trained with larger image obviously trains slower, but apart from that, the generated image and latent space shows little difference.

2.2 Attempt #2

We also tried to train a VAE with lower latent z dimension to study how dimension of latent space affect the generated output image. We found that dimension of z is very important and should be less than the dimension of input image. With $z \geq 64$, the model is over fitting and it is memorizing the image data as shown in figure 4a. With $z = 32$, the latent space is able to learn features that capture meaningful variation exists in the data, such as skin color and emotions. As shown in Figure 4b, the latent vector is able to capture the different skin tone from different people.

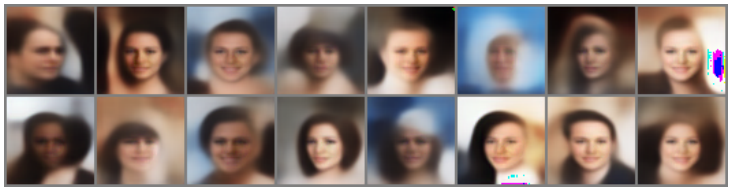
(a) Overfitted sample

(b) Normal sample

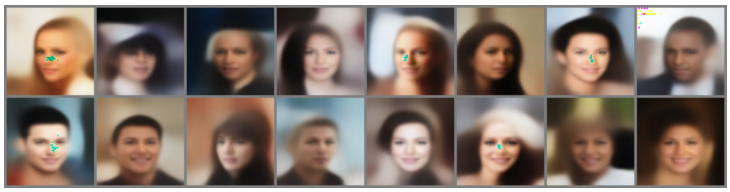
Figure 4: Comparison of VAE latent space

2.3 Generated Samples

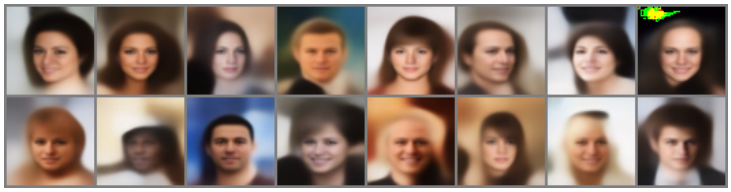
Generally, the images generated by VAE has blurry backgrounds as shown in Figure 5.



(a) Epoch 1



(b) Epoch 5



(c) Epoch 10

Figure 5: Generated Samples of VAE

3 Generative Adversarial Networks

Similar to VAE, GANs typically consist of two network, but with different functions. The generator and the discriminator. The role of the generator is to generate 'fake' images as real as possible whereas the role of the discriminator is to detect the validity of the generated image. The discriminator here is like a support role to help to train the network. After finish training, the discriminator is discarded.

3.1 Deep Convolutional Generative Adversarial Network (DCGAN)

DCGan [6] is a variant of the GAN network with CNN layers in the discriminator and generator. However, the network is limited to be only 64 x 64 x 3 input dimension. Both network are using a fixed learning rate ranging from $1e-3$ to $1e-5$ and trained for 30 epoch, We implemented a DCGan with the architecture shown in Figure 6.

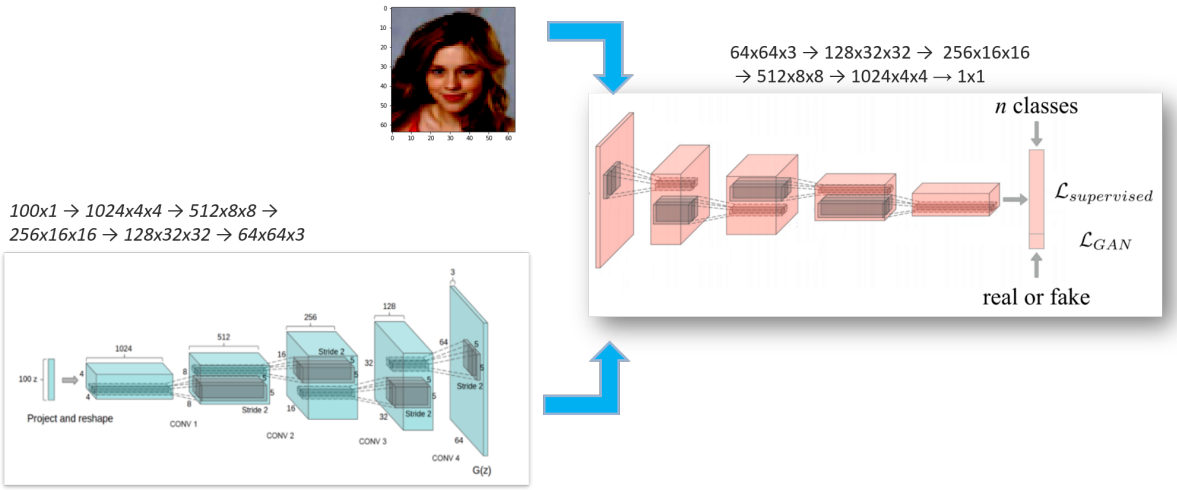


Figure 6: DCGan Architecture

Before starting the training, we use a smaller sample size to trial run the training for 3 epochs to select the best learning rate. If the learning rate is too big, the model loss oscillates a lot and we found that $\text{lr} = 2e-3$ gives a stable training. For the optimizer function, Adam is used with the momentum set to 0.5 because the suggested value of 0.9 results in oscillation and instability.

In Figure 7, the diagram shows the generator network's loss is decreasing whereas the discriminator network loss is increasing. The plotted graph shows that as the training epoch increases, the generator loss is decreasing while the discriminator loss is increasing. This indicates the generator is able to generate images that are capable of confuse the discriminator.

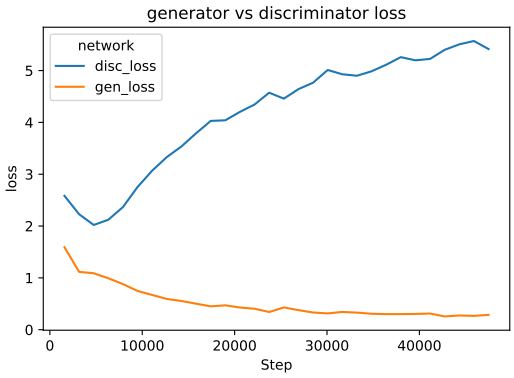
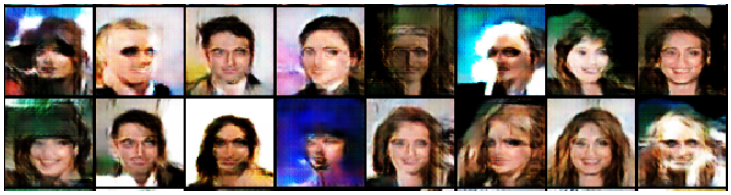


Figure 7: Gan loss by epoch

Figure 8 demonstrates the images generated by the GAN model. We can see that the image are quite good in epoch 30 as shown in Figure 8c and 8b, labelled in yellow.



(a) Epoch 0

(b) Epoch 10

(c) Epoch 30

Figure 8: GANs generated faces

We also found that it is harder to generate images of dark skin color as shown in Figure 8c labelled in red. This maybe caused by the imbalance dataset with lesser faces of other skin color.

Our finding is also supported by this previous studies [3] where face with other skin colors often experience mode collapse, especially in the CelebA dataset has only 10% of Black faces and White faces are at about 80% as shown in Figure 9. This indirectly causes the discriminator to reject Black faces output and the generator will over optimizes for the discriminator to find the most plausible output (in this case are White faces) to cheat the discriminator.

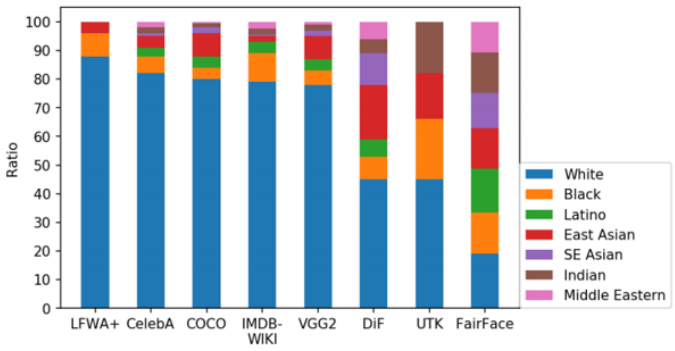
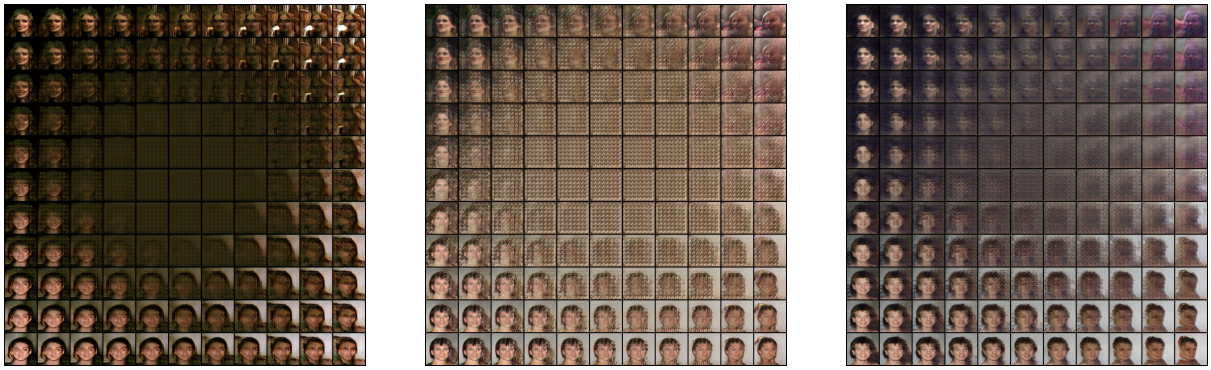


Figure 9: Racial Composition in face datasets [3]

3.2 Analysis of Latent Space

The latent space image for epoch 1, 5 and 10 is generated. For analysis we compared the first and last epoch to understand what the network has encoded.



(a) Epoch 1 (b) Epoch 10 (c) Epoch 30

Figure 10: Latent Space Interpolated

The generator learns to map points into the latent space with specific output images through training. Figure 10a shows the latent space mapped before training and Figure 10c shows the latent space mapped after training. For example, in Figure 10, the model tries to encode the direction of the face facing (top: facing left, bottom right: facing right).

3.3 Vector Math

Another analysis we did is try to analyse latent vector z using vector math as shown in Figure 11. We used a random noise sampled from the $X \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$. and feed into the generator to generate n vectors. Then, we carry out vector arithmetic operation to the n vectors and plot the image using matplotlib. For illustration purpose, We did two examples to demonstrate the vector math.

- The first row is the results of subtracting a smiling man from the smiling woman, we get is a neutral woman.
- The second row is the results of three operation, where a woman with neutral expression and black hair is first added to a blonde with neutral expression, and then the woman with smile and black hair is subtracted from the results . We get samples of blonde with neutral expression.

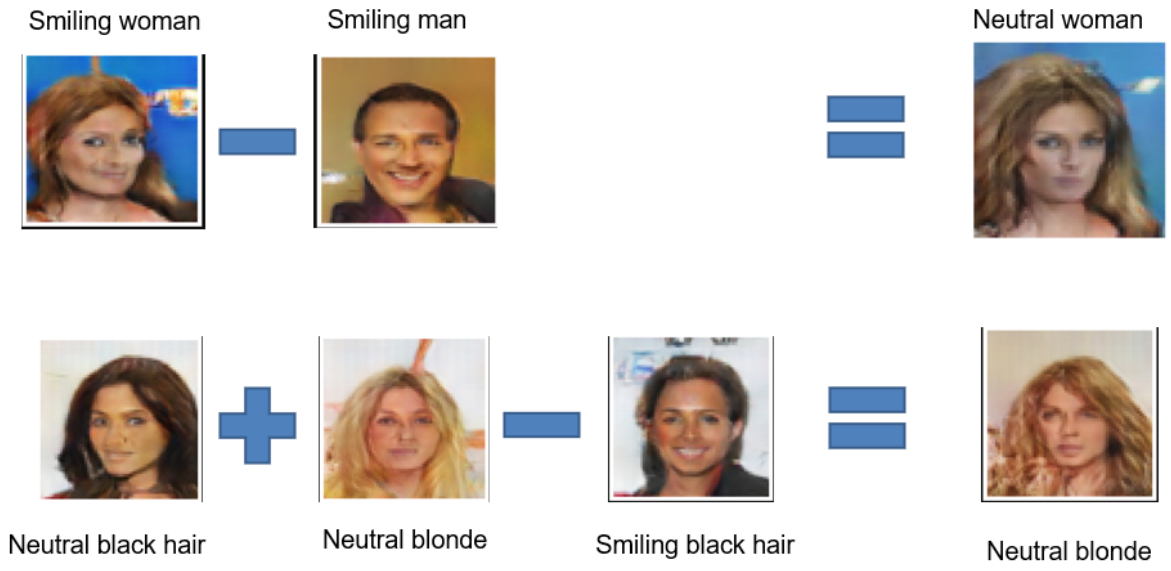


Figure 11: Interpretable Vector Math

4 Comparison

4.1 Training Process

GANs is a implicit density generative model while VAEs is a approximate density estimation model. The main difference between VAEs and GANs is their learning process.

VAE get to pick the features themselves. VAEs tried to minimize a loss of reproducing a certain image with the important features, which the analogy is like after looking at a image, then the model tries to redraw the original image without looking. However, the decoder has no true generative capabilities.

GANs in the other hand is a min-max game where training doesn't involve MLE. The training of a GAN models simultaneously involved two network, which are Generator and Discriminator. The Generator is trained to generate artificial samples from noise, looking as real as possible; and the Discriminator tries to distinguish them from real samples. Moreover, GANs are good at capturing mode of distribution and . However, 'Oscillation' may happen where the model trains for a very long time and even results in failure to converge.

During training, VAEs directly "sees" the sample, so it is prone to overfitting where GANs are more robust to overfitting because it does not "sees" the samples directly

4.2 Latent Interpolation

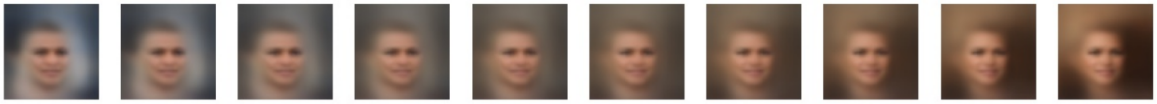
In the latent space of generative models, each feature is actually a sliding scale between two distinct versions of a feature, e.g. male/female for faces shown in Figure 12, or wide/thin brushstroke for MNIST digits.

Figure 12 shows the comparison of interpolation of two latent points which indicates male and female.

4.3 Generated Samples

So, the VAE generally has blurry background as shown in Figure 13a as it focus on face features while GANs reconstruct the face including the background as shown in Figure

(a) Epoch 1

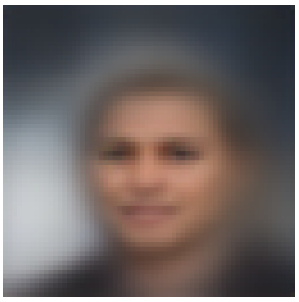


(b) Epoch 10

Figure 12: Latent Space Interpolated from male to female latent points

13b.

(a) DCGAN



(b) VAE

Figure 13: Generated sample by trained model

5 Conclusion

In this image generation task, we implemented two types of generative model, GANs and VAE to generate faces using the CelebA dataset. Besides, we also compared and analysed the two models in terms of training process, vector math, latent space interpolation, and generated samples. Last but not least, we also find that imbalance dataset will affect the sampling of models, which leads to mode collapse problem.

References

[1] URL https://www.programmersought.com/article/72315707422/.

[2] William Falcon and Kyunghyun Cho. A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104*, 2020.

[3] Kimmo Karkkainen and Jungseock Joo. Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1548–1558, 2021.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[5] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.