Jay Buensuceso & Pratheek Sankeshi
COGS 185
Tu
16 Jun 2023

Abstract:

When it comes to image generation, there are multiple models that can be utilized. One model is a Deep Convolutional Generative Adversarial Network (DCGAN) and another is Variational Autoencoders (VAE). Although similar, the two work in differing ways, a DCGAN works through a generator and discriminator where The generator takes random noise as input and generates synthetic samples, while the discriminator tries to distinguish between real and generated samples. The generator and discriminator are trained simultaneously in an adversarial manner, where the generator tries to produce more realistic samples to fool the discriminator. Whereas a VAE utilizes probabilistic models that aim to learn the underlying distribution of the input data and generate new samples from that distribution. In this report, the two models are compared on two datasets, an anime face dataset and an animal faces dataset. Alongside this, a set of hyperparameters will be tuned in each test, with the aim to examine how each model reacts to the differing hyperparameters. Our findings revealed the unique advantages and trade-offs between DCGANs and VAEs, contributing to a better understanding of their applicability in different generative tasks.

Introduction:

Generative models have become a critical research area in machine learning due to their ability to simulate and generate new data instances that mimic the structure and distribution of the training data. Among various generative models, Deep Convolutional Generative Adversarial Networks (DCGANs) [1] and Variational Autoencoders (VAEs) [2] have shown promising results. DCGANs leverage adversarial training methods for generating new data instances, while VAEs rely on a probabilistic graphical model with an underlying latent space. Both techniques provide intriguing solutions for generating new instances but exhibit different strengths and weaknesses, which warrant further investigation.

This study investigates the application of DCGANs and VAEs on two distinct datasets: anime faces and animal faces. These datasets present unique challenges such as high dimensional data and a significant degree of intra-class variations, making them an interesting case study for generative models. We perform extensive experiments to evaluate the effectiveness of these models, comparing their performance based on image quality, diversity of generated samples, and training stability.

In the broader context, this study contributes to our understanding of generative models, particularly DCGANs and VAEs, in dealing with complex, high-dimensional data. The findings provide useful insights for selecting the most suitable generative models based on the characteristics of the dataset and the specific requirements of the task at hand.

Method/Architecture Description:

We used two separate datasets for this study: an anime face dataset and an animal face dataset. The anime face dataset consists of several thousand images of anime faces, all standardized to a certain size and color depth. The animal face dataset comprises a diverse range of animal species, also standardized in terms of image size and color depth.

Our first method is Deep Convolutional Generative Adversarial Networks or DCGAN for short. Compared to VAE's tensorflow, this model was built using Pytorch in order to make use of both deep learning architectures. As the name suggests, the model trains in an adversarial manner, with a generator and discriminator, both of which are trained with different objectives, being paired together to generate new images. The generator network takes a random noise vector as input and produces an image, whereas the discriminator network takes an image as input and outputs a binary classification, determining whether the input image is real (from the dataset) or fake (generated by the generator). Each layer of the network uses either ReLu or leaky ReLu as its activation function, depending on the hyperparameter tuning. ReLu, is defined as:

$$f(x) = max(0, x) \; where \; x \; is \; the \; input \; of \; the \; function.$$

Although ReLu is computationally efficient and can accelerate training because it does not involve complex mathematical operations, it can still suffer from a concept known as "dying reLU" where a large number of neurons become inactive and the model stops training. TO mitigate this, another activation function known as leaky ReLU is introduced. Leaky ReLU is defined as:

$$f(x) = max(\alpha x, x) \; where \; \alpha \; is \; defined \; as \; a \; small \; positive \; constant$$

By introducing a small negative slope for negative input values, it allows information to pass to inactive neurons, allowing them to learn - successfully mitigating the"dying ReLU" problem. This also has the added benefit of converging faster and improved performance in certain scenarios.

In the case of the generator and discriminator, the generator uses either ReLU or leaky ReLU, while the discriminator utilizes a leaky ReLU. This is based on pytorch's model for

DCGANs authored by Nathan Inkawhich. The implementation of DCGAN also measures the loss of the generator and discriminator along with an attempt of an implementation of FID heavily based upon mseitzer's implementation of pytorch-fid.

Our secondary method of choice was the Variational Autoencoder (VAE) implemented with TensorFlow. In contrast to the adversarial training approach used in DCGANs with PyTorch, VAEs work on the principle of encoding and decoding, and they are known for their ability to learn a compact, continuous representation of the input data. The VAE consists of an encoder, a decoder, and a loss function. The encoder network takes an image as input and produces a distribution in latent space, which is usually a multidimensional Gaussian. The decoder network takes a point sampled from this distribution as input and generates an image. The loss function of VAE is comprised of two parts: the reconstruction loss and the KL-divergence, which ensures that the latent space has good properties. The activation function used in this network is either ReLu or leaky ReLu, depending upon the hyperparameters. As previously described, ReLu is computationally efficient but can suffer from the "dying ReLu" problem. To counteract this, we again use leaky ReLU as an alternative.

In our TensorFlow VAE model, both the encoder and decoder utilize either ReLu or leaky ReLu, depending on the specific implementation details. The TensorFlow VAE model is built based on the implementation by François Chollet in the Keras examples. We measure the Fréchet Inception Distance (FID) to evaluate the quality of the generated images, which is inspired by the mseitzer's PyTorch implementation. We also utilize the Inception Score (IS) to evaluate the quality of the generated images in our TensorFlow Variational Autoencoder (VAE) model. Both the Fréchet Inception Distance (FID) score and the Inception Score (IS) are critical tools in our model's evaluation arsenal. They allow us to comprehensively analyze the quality of the images generated by our TensorFlow Variational Autoencoder (VAE).

The Inception Score, proposed by Salimans and others, evaluates the image quality produced by generative models based on two central tenets. Firstly, it measures the diversity of images by examining the entropy of the predicted classes, with the assumption that a model producing varied images is superior. Secondly, it assesses the clarity of generated images, insisting that they should distinctly represent a particular class, rather than being an amalgamation of different classes. This score is derived using a pre-trained Inception Network, which classifies the generated images, with the resulting class distribution used to calculate the score. The calculation is complex, utilizing both the entropy of predicted class distribution for individual images and for the overall generated image dataset. On the other hand, the FID, introduced by Heusel and others, provides an evaluation of image quality by comparing the statistical distribution of generated images with the distribution of real images. It uses the Inception Network for feature extraction, then computes the Fréchet distance between two multivariate Gaussian distributions, which correspond to the feature representations of real and

generated data. Hence, the FID score serves as a measure of the distance between these two distributions in the feature space, providing a comprehensive evaluation of the quality and diversity of generated images. Lower FID scores signify higher similarity between the datasets, implying superior quality of the generated images.

In essence, both the Inception Score and Fréchet Inception Distance are crucial in evaluating the performance of our generative model. However, they approach this task from different perspectives: the Inception Score prioritizes image diversity and clarity, while the FID quantifies the statistical similarity between real and generated images.

Experiment:

Training and evaluating generative models on anime and animal face datasets serve several purposes and provide unique challenges that can help us understand the performance and limits of these models. Anime faces and animal faces are both diverse and complex. On one hand, Anime faces often contain exaggerated features and a wide range of artistic styles, whereas animal faces encompass a broad spectrum of species, each with distinct characteristics. This level of diversity presents a challenge for generative models to capture and reproduce, providing a robust test of their capacity to learn varied and complex data distributions.

The generated results from both these datasets can be easily visually inspected as humans are naturally adept at recognizing faces, including anime and animal faces, so it's easy to tell when generated images are not plausible.

Additionally, each dataset has unique applications. For anime faces, potential applications could include animation, video game design, or other digital art projects where new, unique anime faces might be needed. For animal faces, potential applications could extend to biological research or conservation efforts, where generating images of animals could be used for educational purposes, simulations, or data augmentation in other machine learning tasks..

Our training procedure involved using Adam and SGD optimizers with learning rate decay. The Pytorch DCGAN model was trained on a NVIDIA T4 and V100 GPUs for a predetermined number of epochs until the quality of generated samples saturated. The Tensorflow VAE model was trained on a A100 GPU on Google Colab. Due to the hardware constraints we were forced to run our data on Google Colab for the free GPU usage and eventually buy credits in order to let the Tensorflow VAE models run faster.

The DCGAN model was used in seven tests, each of which having a different set of hyperparameters. Each test goes as follows:

For the anime face dataset:

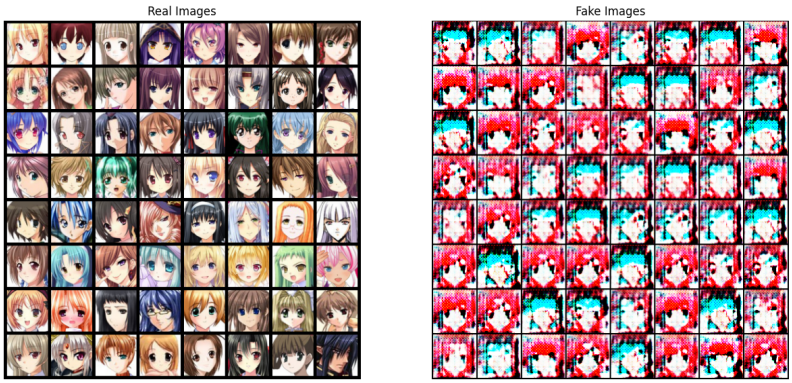
- ReLU generator with 0.001 learning rate at 5 epoch
- ReLU generator with 0.01 learning rate at 5 epoch
- Leaky ReLU generator with 0.001 learning rate at 5 epoch
- Leaky ReLU generator with 0.01 learning rate at 5 epoch
- Leaky ReLU generator with 0.001 learning rate at 20 epoch

For the animal face dataset:

- Leaky ReLU generator with 0.0001 learning rate at 10 epoch
- Leaky ReLU generator with 0.0001 learning rate at 20 epoch

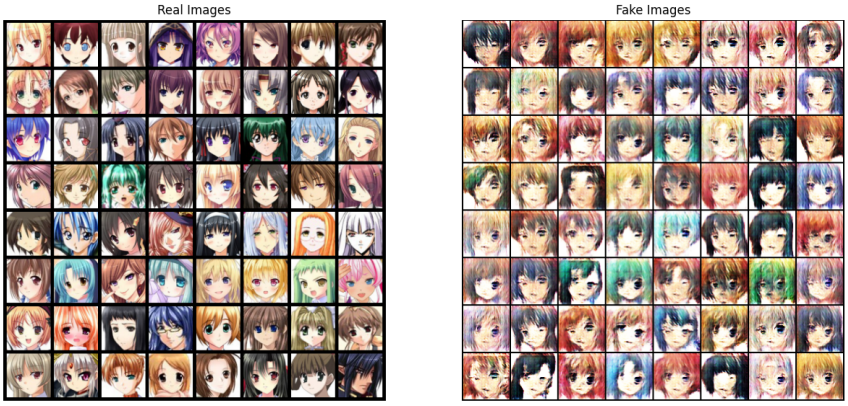
1) 0.01 Learning rate with Leaky ReLU activation function on generator:

2) 0.001 Learning Rate with Leaky ReLU activation function on generator:

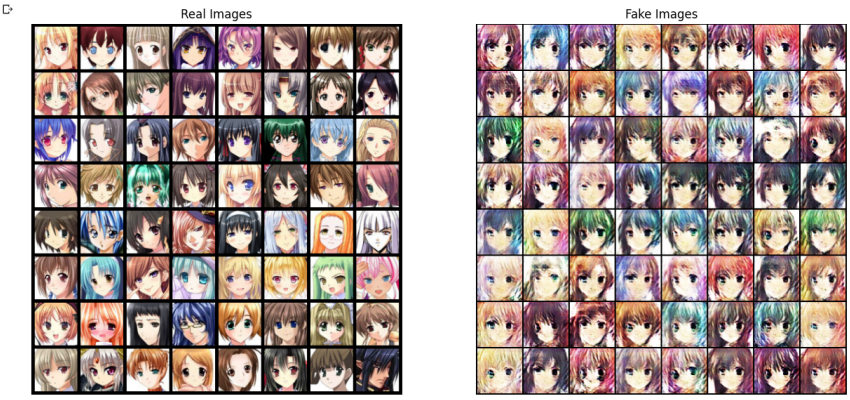


3) 0.01 Learning rate with ReLU activation function on generator:

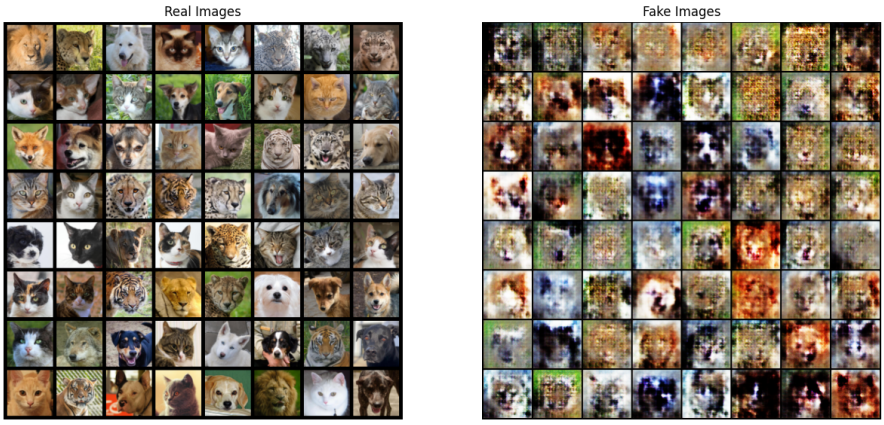
4) 0.001 Learning Rate with ReLU activation function on generator:



5) 0.001 Learning Rate with ReLU activation function and 30 epoch:



6) ReLU generator with 0.0001 learning rate at 10 epoch (animal dataset)



7) ReLU generator with 0.0001 learning rate at 20 epoch (animal dataset)

The VAE model was used in 6 tests, each of which having a different set of hyperparameters. Each test goes as follows:

For the anime face dataset:

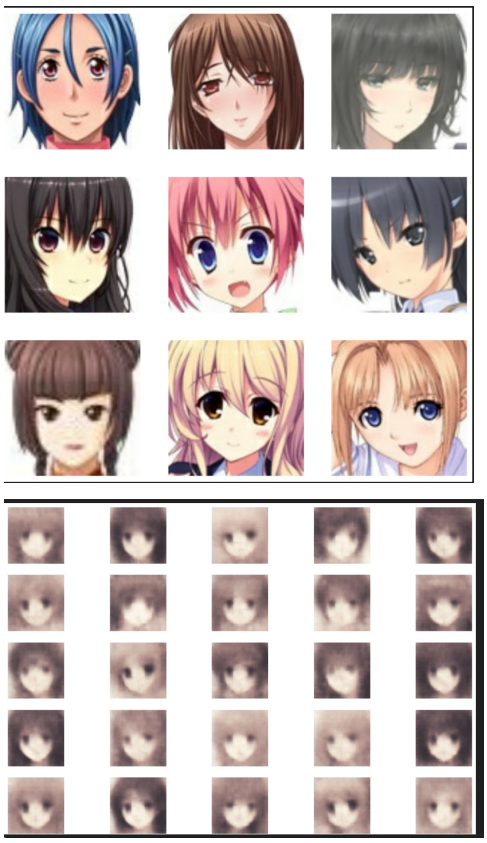
- ReLU Encoder, ReLu Decoder with 0.001 learning rate at 5 epoch
- ReLU Encoder, ReLu Decoder with 0.005 learning rate at 5 epoch
- Leaky ReLU Encoder, Leaky ReLu Decoder with 0.001 learning rate at 5 epoch
- Leaky ReLU Encoder, Leaky ReLu Decoder with 0.005 learning rate at 5 epoch

For the animal face dataset:

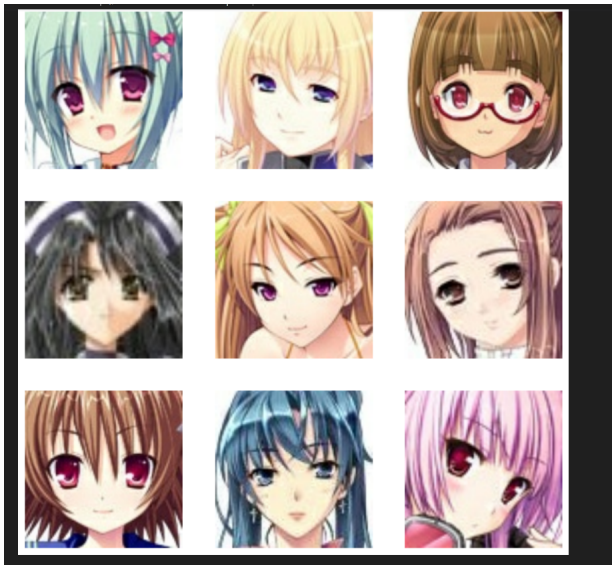
- ReLu Encoder, ReLu Decoder generator with 0.0001 learning rate at 5 epoch
- Leaky ReLu Encoder, Leaky ReLu Decoder with 0.0001 learning rate at 5 epoch

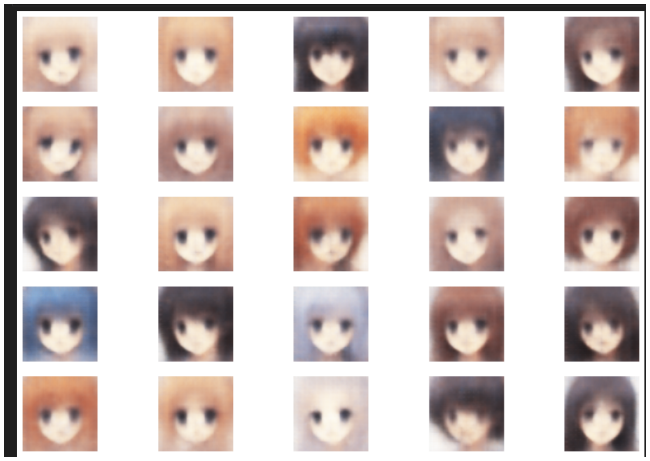
Format of pictures are original pictures first and then the generated:

8) 0.01 Learning rate with ReLU activation function for Encoder and Decoder



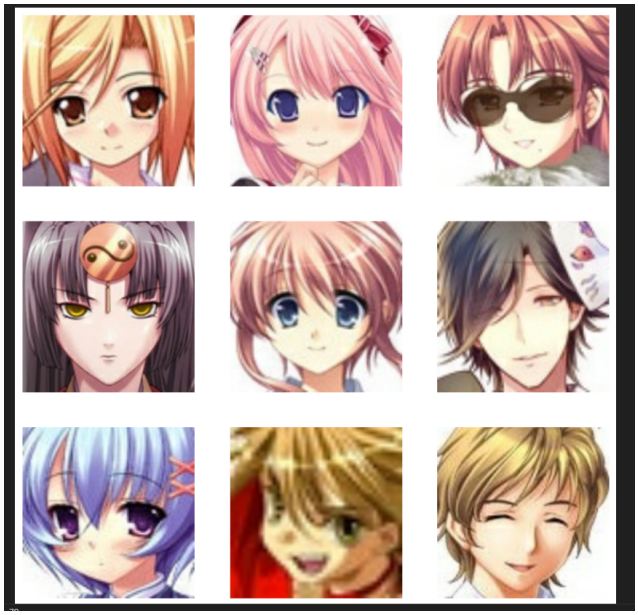
9) 0.0005 Learning Rate with ReLU activation function on generator:

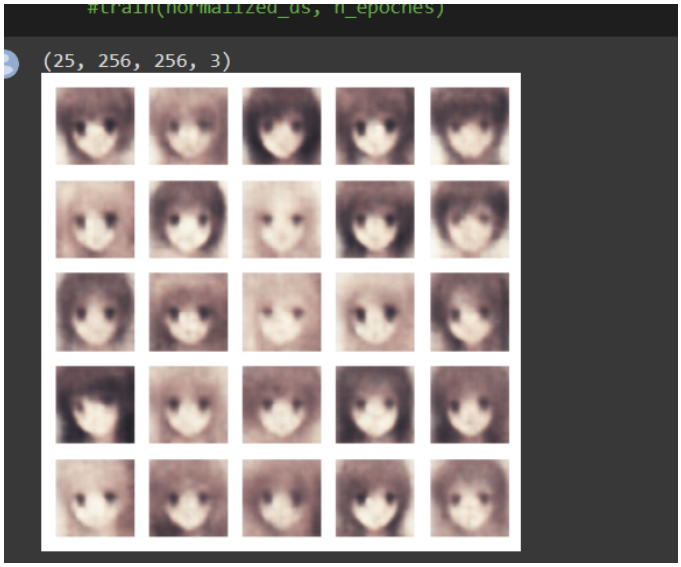




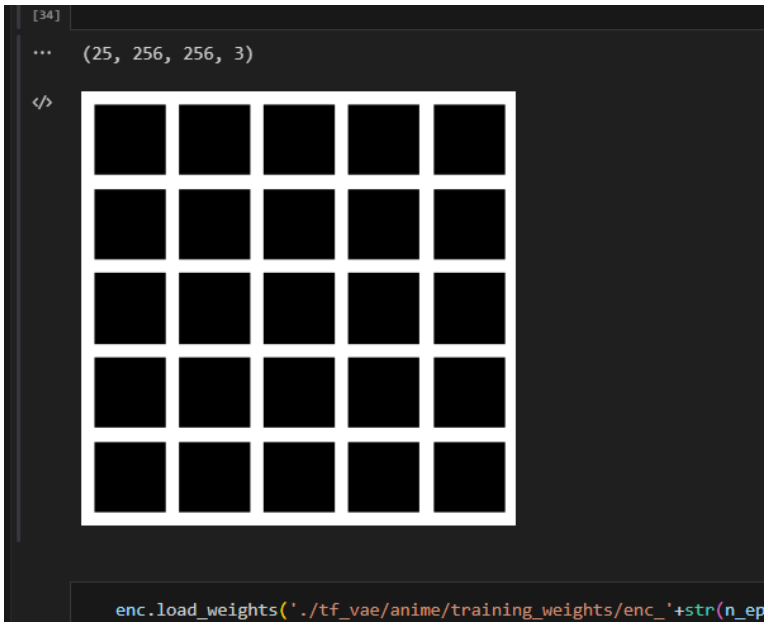
10) 0.0005 Learning rate with Leaky ReLU activation function for Encoder and Decoder:
Data was corrupted but results were similar to the 0.001 Learning rate for Leaky ReLU

11) 0.001 Learning Rate with Leaku ReLU activation function on Encoder and Decoder:

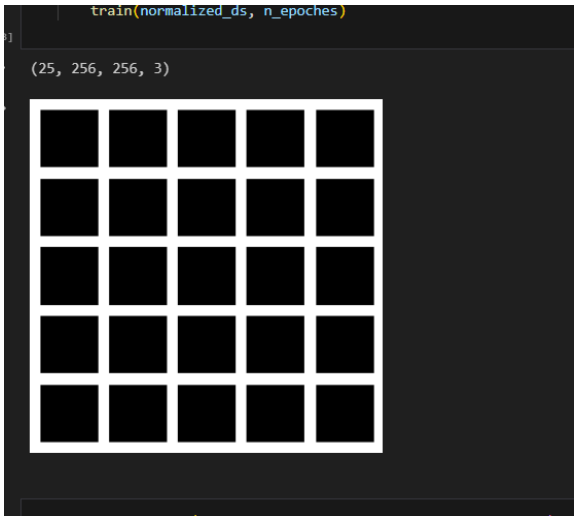




12) 0.001 Learning Rate with ReLU activation function and 5 epoch (animal dataset): Images wouldn't generate for any of the animal dataset and due to training constraints and end of credits couldn't train anymore:



13) 0.01 Learning Rate with Leaky ReLU activation function and 5 epoch (animal dataset):



Discussion:

As can be seen from figures 1 through 7, a variety of hyperparameters were tested - namely learning rate, epochs, and activation functions. On the anime dataset, a generator with the ReLU activation function performs better visually than a generator with the Leaky ReLU activation function. Additionally, as we increase in epoch amount, the model performs better as more iterations are conducted between the generator and discriminator. In the context of vusal fidelity for generated artwork, a model that runs at 30 epoch is move visually appealing compared to models running at a lower epoch rate.

For figures 6 and 7 we tested the DCGAN on an animal face dataset. After preliminary tests show that a larger number learning rate (i.e. 0.01) led to static unusable data, a decision was made to compare 10 and 20 epoch generated data. Figure 6 shows the model running at 10 epoch

and visually some of the images are hard to distinguish from one another. A model running at 20 epoch on the other hand, shows more distinctive animals being generated such as wolves and cheetahs.

Additionally, though an attempt was made to implement The Frechet Inception Distance (FID) score to the DCGAN model, many factors led to inconclusive results that were extremely incorrect from the expected outcome. One such example is the Leaky ReLU tests showed a lower FID score compared to the ReLU tests even though the error metrics and visual examination show ReLU tests performing better than its counterpart.

As can be seen from figures 8-14, a variety of hyperparameters were tested - namely learning rate, epochs, and activation functions. Some of the hiccups in the VAE models were that we couldn't test different optimizers and layers without causing the model to no longer converge, an issue we would like to work on in the future. Results for VAE were similar to DCGAN however some key points we noticed were that VAE was noticeably slower to train to the point we had to buy credits on Google Colab to train faster and not run out of time and usage limit. Furthermore, everything except for Adam as an optimizer didn't work, something we would like to fury explore in the future.

Conclusion

The results show that the choice of hyperparameters significantly impacts the performance of the models. On the anime face dataset, the generator with the ReLU activation function produces visually better results compared to the generator with leaky ReLU. Increasing the number of training epochs also improves the visual quality of generated samples. On the animal face dataset, we show that a generator with leaky ReLU and a learning rate of 0.0001 performs well, particularly when trained for 20 epochs. The generated samples exhibit more distinct and recognizable animal features. In the event that an experiment of this caliber is to be repeated, a thorough check of the Leaky ReLU implementation should be conducted as in theory it should perform as well as ReLU however the findings show that it did not perform well when compared to its counterpart.

The findings highlight the advantages and trade-offs between DCGANs and VAEs in image generation tasks. DCGANs excel in capturing visual fidelity and generating diverse samples, while VAEs focus on learning the underlying distribution of the data. Overall, This study contributes to a better understanding of the applicability of these models in different generative tasks, providing insights for selecting the most suitable approach based on the characteristics of the dataset and specific requirements.

References

Churchill, Spencer. “Anime Face Dataset.” *Kaggle*, 13 Oct. 2019, www.kaggle.com/datasets/splcher/animefacedataset.

DCGAN Tutorial — PYTORCH Tutorials 2.0.1+CU117 Documentation, pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html. Accessed 17 June 2023.

Kingma, Diederik P, and Max Welling. “Auto-Encoding Variational Bayes.” *arXiv.Org*, 10 Dec. 2022, arxiv.org/abs/1312.6114.

Larxel. “Animal Faces.” *Kaggle*, 22 May 2020, www.kaggle.com/datasets/andrewmvd/animal-faces.

Mseitzer. “Mseitzer/Pytorch-FID: Compute FID Scores with Pytorch.” *GitHub*, github.com/mseitzer/pytorch-fid. Accessed 16 June 2023.

Radford, Alec, et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” *arXiv.Org*, 7 Jan. 2016, arxiv.org/abs/1511.06434.

Salimans, Tim, et al. “Improved Techniques for Training Gans.” *arXiv.Org*, 10 June 2016, arxiv.org/abs/1606.03498.