DCGANs Evaluation Measurement

Bradley Hedges
Department of Mathematics
University of Waterloo
Waterloo, Canada
bshedges@uwaterloo.ca

Dongni Xie
Department of Mathematics
University of Waterloo
Waterloo, Canada
d38xie@uwaterloo.ca

Xi Chen
Department of Mathematics
University of Waterloo
Waterloo, Canada
x688chen@uwaterloo.ca

***Abstract*—Generative models, which are known as generative adversarial networks (GAN), have become popular in the fields of computer version and machine learning in recent years. In this project, we built a DCGAN architecture using TensorFlow library in Google Colab using CIFAR-10 dataset. Afterwards, we evaluated the performance of the DCGAN model in terms of the quality of generated images. We outlined some GAN evaluation measures including one manual experiment and two quantitative measures with some creative approaches, and analyze their strengths and limitations respectively.**

***Index Terms*—Generative Adversarial Network, Deep Learning, Evaluation, Convolutional Neural Networks**

I. An Introduction to *DCGANs*

In recent history, a new technique for generating images has been introduced, called a *Deep Convolutional Generative Adversarial Network* (henceforth referred to as *DCGAN*). DCGANs use ideas from *Convolutional Neural Network* (or CNN) architecture to stabilize the training of the otherwise notoriously unstable GAN.

In general, GANs consist of two competing elements: a *generator* network and a *discriminator* network. To train a GAN, the generator is given a random vector in the *latent* space, and outputs a candidate image. The generated image is then sent to the discriminator, where it attempts to distinguish the image from a dataset of true images (for example, CIFAR-10). In essence, the generator and discriminator play a "game" where they try to outwit one another; the generator attempts to generate images similar to the true distribution, and the discriminator attempts to identify the fakes. While the entire GAN is being trained in this way, the discriminator is also being trained with gradient descent on a sample of generated images and the true dataset to correctly classify the images. Ideally, the training continues until the generator is able to create images that a well-trained discriminator cannot accurately identify.

GANs on their own come with some common issues, the foremost being that they are often unstable, and lead to totally nonsensical outputs. To combat this, Radford et. al. introduced a new category of GAN, called the DCGAN. In a typical GAN architecture, the generator consists of sequences of 1D, fully-connected layers. The DCGAN architecture, which we will investigate in a later section, uses a *convolutional* layering technique to up-sample (in the case of the generator) and down-sample (with the discriminator). The term convolution comes from each layer having some matrix convolution of the information from the previous layer. It has been shown that this technique significantly stabilizes network training.

One caveat in the construction of the DCGAN is the lack of a true objective function. In most neural networks, the objective function acts to justify how well a network is doing at its given job. Especially in supervised learning, the objective function gives a useful metric for measuring success. The DCGAN model does not have such an obvious value for measuring success. Rather, its objective function measured how well the generator is fooling the discriminator. Since the discriminator and generator form a closed loop where each trains the other, this measure is effectively irrelevant aside from making sure the ouputs are not blowing up.The purpose of this paper is to investigate some of the pre-existing

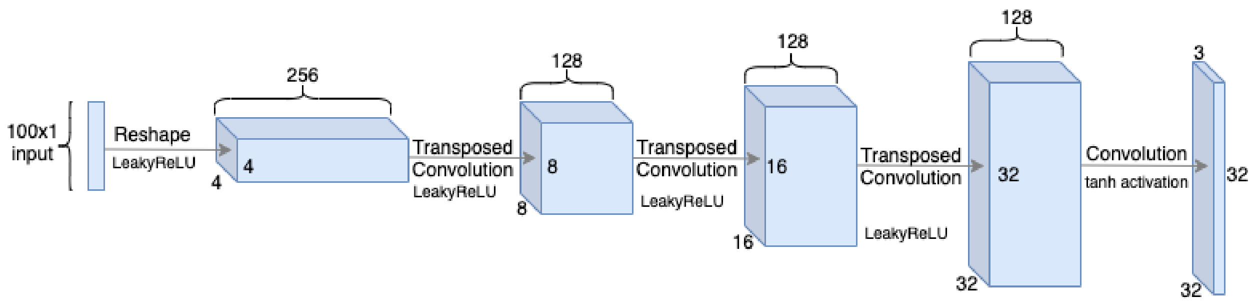

Fig. 1: DCGAN generator architecture for use with CIFAR-10 dataset. A LeakyReLU activation function was used between convolutions, and the final layer has a tanh activation function to produce output in the range $(-1, 1)$. A CIFAR-10 image can be represented as a $32 \times 32 \times 3$ matrix. Inspiration for this image comes from [1].

techniques to quantify discriminator success, as well as pose our own techniques.

II. DCGAN Architecture

As explained previously, a standard DCGAN consists of a discriminator and a generator. In this section, we will outline the general structure of each, and how we have modified it in our implementation. Due to relatively expensive training cost, we opted to train on two datasets: *MNIST* and *CIFAR-10*. For the purpose of this paper, we will outline our chosen model for CIFAR-10, as it is more illuminating than our MNIST model. Figure 1 shows the how our convolutions change the shape of each layer. The generator model takes in a 100×1 latent vector, adds 3 transposed convolutional layers, each with a *LeakyReLU* activation function. There is then one convolutional layer to have the appropriate output shape with a *tanh* activation function to produce outputs in $[-1, 1]$. The discriminator model takes in the $32 \times 32 \times 3$ image, adds 4 convolutional layers with *LeakyReLU* activation function, adds dropout to the penultimate layer, and finally produces a single unit output in $[0, 1]$. Beyond this, we also have hyperparameters such as initialized weights, learning rate and activation function parameters; these were chosen and modified according to [1].

III. Image Generation

Figure 2 shows 5 images generated by passing random points in latent space through a DCGAN that was trained on CIFAR-10 .

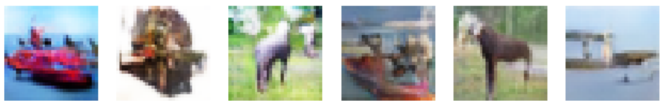
Fig. 2: Fake CIFAR-10 images constructed by a DCGAN

Since the CIFAR-10 dataset contains 10 classes of image including horses, dogs, trucks, boats and airplanes. As we can see from the figure, the DCGAN was able to construct images that resemble boats (images 1 and 4), dogs (image 3), trucks (image 2), horses (image 5) and airplanes (image 6). Despite the apparent success of the GAN, it also produced formless shapes, as shown in figure 3.

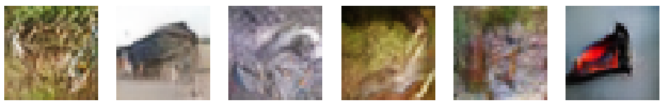
Fig. 3: Formless CIFAR-10 images constructed by a DCGAN

Furthermore, the DCGAN produced formless noise much more readily than clear, identifiable images. The formless noise does have an explanation, however. We can view the latent space as an compact encoding of the $32 \times 32 \times 3$ images, much like an autoencoder. In an autoencoder, similar data is grouped closer together in the latent space. We can then consider the formless images to be some combination of the latent vectors of more well-defined images.

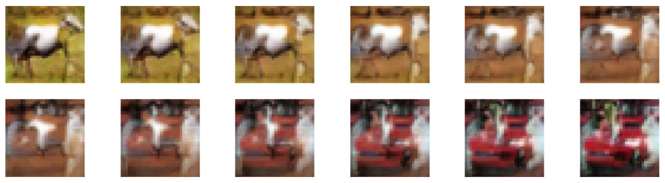
Fig. 4: Generator output from stepping between vectors in latent space

Figure 4 gives an example of the generator producing unidentifiable images. The figure is the result of stepping in latent space from a generated image resembling a four-legged animal to a generated image resembling a red car. In the intermediate steps, the generator outputs an image that looks only like noise, with all the identifying features of a four-legged mammal or car being muddied together. This smooth transition from one image to the other suggests that the DCGAN was successful in learning a true representation of the images.

IV. Evaluating Success

As explained earlier, there is no ubiquitous method for evaluating the success of a DCGAN in a meaningful way. In this section we will outline some techniques for measuring success, including an experiment of our own design.

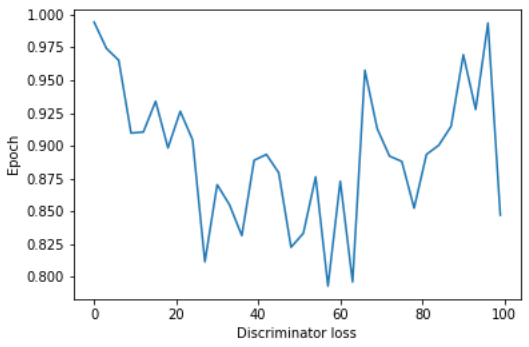

Fig. 5: Discriminator loss over the first 100 epochs of training a DCGAN on CIFAR-10.

Figure 5 illustrates the fluctuating nature of the discriminator loss. Since the generator and discriminator are training each other, the loss oscillates despite the DCGAN getting objectively better at generating and identifying images. We now propose the following techniques to measure the quality of DCGANs.

A. An Experiment of Our Own Design

We will first outline our own experiment. The experiment was initialized as follows:

1) Train the DCGAN model on the appropriate dataset, until we decide the images look "good enough".

2) Randomly choose 25 true images from the dataset.
3) Generate 25 random vectors in latent space.
4) Run the random vectors through the generator to find their corresponding fake image.
5) If a fake image isn't meaningful, throw it out and pick a different vector.
6) Collect the 49 images, and shuffle them.

Choosing random vectors (and therefore random generated images) is perhaps the most important step in this initialization step. Cherry-picking images that "look real"is perhaps enticing, but then the images do not represent the generator; rather, they represent a small selection of points that are near real images in latent space.

At this point, we have a list of 49 images, approximately half real and half generated, and their order randomized. An example can be seen in figure 6. With this, we can introduce our evaluation protocol.

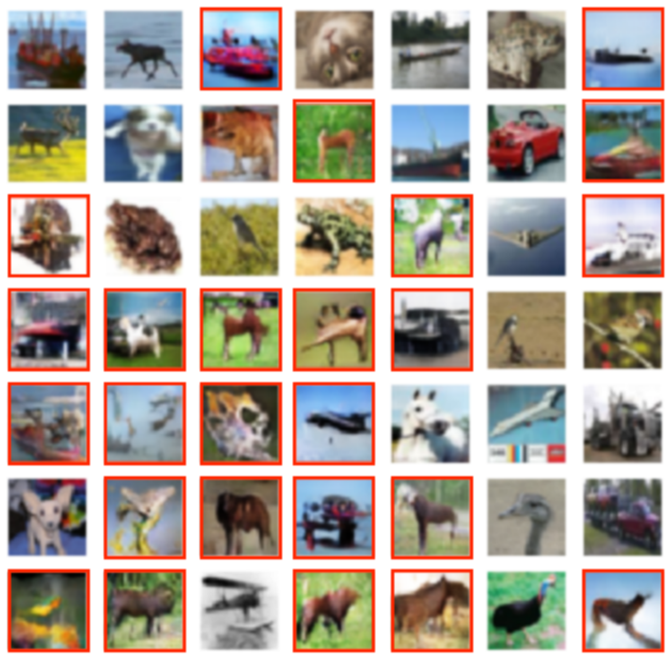
Fig. 6: An example of 49 images collected for the experiment (images surrounded by a red box are generated).

Our first measure of the quality of a generator is how frequently it can fool people. Given a list of generated images, the ratio of incorrect guesses is a fair measure of the generators efficiency. The higher the ratio, the better the generator is at tricking people. We chose to add 24 real images to validate how good the individual is at recognizing the real images, as well as to give them a benchmark of how an image should look like. Our experiment is as follows:

1) Give 49 images to the guesser.
2) Specify the guesser to evaluate the images as real or fake as quickly as possible, giving them at most 2 seconds per image. This is because with sufficient investigation, the generated images are obvious.
3) Repeat with two more sets of images. With this we will be able to identify if the guesser improved over time.

The discriminator will be evaluated on the same image sets. If the discriminator is more successful at correctly identifying true images than the humans, we know it is trained well. Accordingly, if the generator is able to deceive the guessers approximately as well as it is able to deceive the discriminator, we can say it has been trained to consistently produce realistic images.

By inviting our friends and families to participate in this designed experiment, we have collected 19 samples. Then we calculate the correct recognition ratio(CRR) by the following formula:

$$CRR = \frac{number\ of\ fake\ image\ recognized}{total\ number\ of\ fake\ image} \tag{1}$$

Each person has three correct recognition ratio(CRR) since they've done the inspection on three sets of pictures. Then we can get an individual correct recognition ratio(ICRR) by taking an average of their three CRR values. We noticed that a person had an individual correct recognition ratio(ICRR) of 0.96. By checking his result, 99% of the images were given a fake image label in his experiment result, so we removed this outlier. Then, by taking an average of all the individual

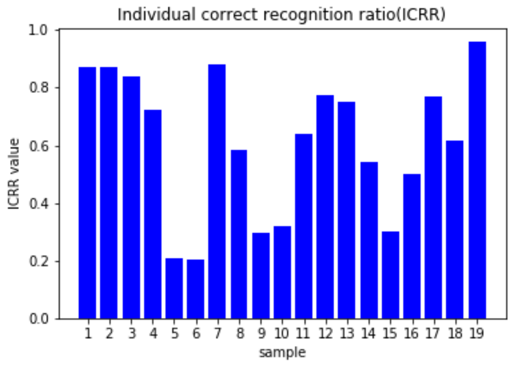

Fig. 7: Sample individual correct recognition ratio(ICRR) values

correct recognition(ICRR):

$$overall\ performance = \frac{sum\ of\ ICRR}{total\ number\ of\ samples} \tag{2}$$

The overall performance of our model is 0.5943, which means approximately 59.43% of the fake images were recognized by our experiment volunteers. Comparing with the performance of our discriminator(87%), which means approximately 87% fake images were recognized, it can be seen that our discriminator outperforms human inspection. This is a satisfying result showing that our generator and discriminator are well-trained.

An interesting fact we found while analyzing the survey data was that people who knew some knowledge about computer vision and neural networks performed better than others without any related knowledge. Therefore, this method is proven biased. It requires some knowledge to judge which images are realistic and some techniques to perform well in order to achieve our target.

In summary, an advantage of this experiment comes in its simplicity; there are no complex equations to implement or expensive computations. However, the experiment is largely subjective and doesn't give much empirical evidence that the DCGAN has been trained well. Furthermore, the experiment doesn't evaluate how well the generated images match the underlying distribution of the true images. Finally, this method is limited in the amount of images that can be reviewed in a reasonable amount of time [2]. With this in mind, we can introduce another, more sophisticated evaluation method.

B. Inception Score (IS)

Another method for GANs performance evaluation currently is Inception Score (IS). Tim Salimans, et al. proposed this method in [4]. The Inception Score method is developed as an attempt to remove the subjective human evaluation of images. During the Inception Score calculating process, numerous pictures are classified by the model. To be more specific, the probability of the image belonging to each class is predicted, and these probabilities are used to summarize into the Inception Score in the end.

To introduce the calculation of Inception Score, we first must investigate the inception v3 Network. Given an image x, the inception v3 Network's work is to output a class label y in the form of a vector of probabilities $p(y|x) \in [0,1]^{number\ of\ classes}$. This vector describes the conditional probability that a given picture belongs to each category. The inception Score(IS) statistic is calculated using the following formula:

$$IS(G) = \exp\left(\mathbb{E}_x D_{KL}(p(y|x))||p(y))\right), \tag{3}$$

where x is the given image, $D_{KL}(p|q)$ describes the KL divergence between the distribution p and q.$p(y|x)$ is the conditional probability. The marginal class distribution is defined as

$$p(y) = \int_x p(y|x)p(x).$$

The score seeks to capture the image quality and diversity of a collection of generated images:

1) Image Quality. The GANs model should be clear rather than vague.i.e the generated picture should be confident to have one clear object. Therefore, $p(y|x)$ should have a low entropy.
2) Image Diversity. The GANs model should generate a variety of different pictures from different classes.

If an image has both satisfying quality and diversity, the IS score should be large. The process we made to evaluate the performance of our GANs using IS is as follows:

1) Generate 10000 pictures using our DCGANs model trained on CIFAR-10.
2) Calculate IS score based on the generated 10000 pictures.

We got a result of 6.5553 for the IS score, which was a good sign indicating our generated pictures had satisfying qualities.

Inception Score(IS) also has some drawbacks. For instance, it is an asymmetric measure, preferring models that generate good objects rather than realistic images, and it is not robust to noise. Thus, an improved method based on IS will be introduced.

C. Frechet Inception Distance (FID)

Proposed as an improved measurement over IS by Martin Heusel, et al. in [3], FID is seen as a better quantitative method to evaluate the quality of generated images. It uses the statistics generated by a GAN and measures the actual distance between raw and generated images. FID tries to compare two multivariate Gaussians from real images and fake images as follows:

$$FID = ||\mu_r - \mu_g||^2 + Tr(\sum_r + \sum_g -2(\sum_r \sum_g)^{\frac{1}{2}}),$$

where $X_r \sim N(\mu_r, \sum_r), X_g \sim N(\mu_g, \sum_g)$, which represent the activation distributions for real and fake images respectively.

Through this measure, FID captures the similarity between two groups of images. A lower FID score indicates that two groups of images have higher similarity, which means our performance of generator model is great on generating high-quality and diversified images by learning from the training data (for example, real images from CIFAR-10). The measuring procedure for FID is as follows:

1) Load Inception v3 model in Keras and captures the activation of feature vector of each image.
2) Remove the top layer of the model and specify the dimension of the input images, and resize the shape of the input from (32,32,3) into (299,299,3) using the Nearest Neighbor Interpolation. After resizing the input shape, scale the image pixel values to fit the input shape for inception model.
3) Calculate mean and covariance of two groups image.
4) Calculate the sum squared difference between means, and square root of product between their covariance.
5) Get FID score

We wanted to evaluate the overall performance for 3000 images across all classes over the first 100 epochs under available GPU, and the result of FID after 100 epochs was 46.3845. Figure 8 shows the existence of oscillation of FID due to some divergence in training, but the overall trend decreased along with the increasing of the number of epochs. We observed that the DCGAN follows the true image distribution well since the FID is relatively low.

Since the training process of DCGANs involves unsupervised learning, it becomes a problem to evaluate how well the DCGANs model performs on the underlying statistics of the training dataset. To tackle this problem and to reveal the DCGANs performance by direct observation, we came up with an innovative way to compare the true CIFAR-10 images and the images regenerated by our DCGANs using underlying

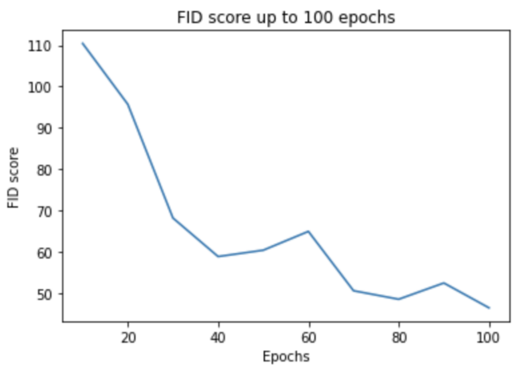

Fig. 8: FID score when training the GANs model

CIFAR-10 data statistics for each class. To be noticed, FID is more robust to noise and performs more diversified images than IS only if we provide enough training data. We used 5000 images to verify a good diversification for each class so that FID would be a good evaluation measurement in terms of image quality and diversity. The detailed process is as follows:

1) The CIFAR-10 data can be split into 10 classes according to the object the image contains. The 10 classes are: *airplane, car, bird, cat, deer, dog, frog, horse, boat* and *truck*. Each class contains exactly 5000 images.
2) Pass the pictures of each class through the reverse generator, which returns the underlying CIFAR-10 data statistics of each class.
3) Pass the statistics getting from the reverse generator through the generator, we can get the regenerated images from our DCGANs.
4) Compute the FID score using the true images and regenerated images. Each class generates a FID score, and the result for each class is shown below in the table 1.

With this procedure, we intend to construct a metric describing how well the DCGAN is able to replicate true images within a specific class, relative to its ability to reconstruct images from other classes. Unfortunately, since our process only compares the classes against each other, we cannot estimate how well the DCGAN represents the statistics of the entire dataset. Instead, our metric describes the DCGAN's relative ability to represent the underlying statistics of each class.

We can observe that some classes perform better than others, such as *airplane*, *cat*, *boat*, *car* and *horse*; some classes perform worse, such as *dog* and *frog*. Figure 9 highlights the generator's ability to reconstruct the different classes. The FID score for class *frog* is particularly high; this is reflected in the frog image being poorly reconstructed. Similarly, the *boat* class has a particularly low FID score, and the corresponding boat image in figure 9 is reconstructed fairly well. According to the table, the generator was able to reconstruct classes *airplane, boat* and *horse* fairly well, and was unable to efficiently reconstruct classes *frog, dog* and *bird*. This is also consistent with our qualitative findings.

TABLE I: general performance for each class under DCGANs

FID for each class	
class 1: airplane	59.4618
class 2: car	65.2227
class 3: bird	76.2951
class 4: cat	58.5620
class 5: deer	76.2495
class 6: dog	82.1838
class 7: frog	85.6249
class 8: horse	65.3323
class 9: boat	48.6967
class 10: truck	71.0192

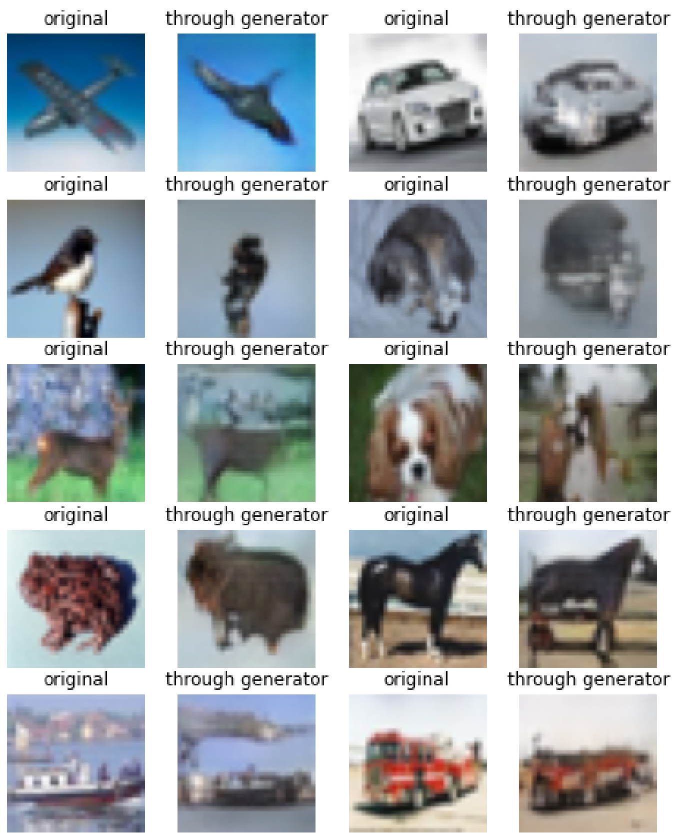

Fig. 9: Examples of real images passes through the reverse generator and back through the generator from each CIFAR-10 class.

V. Conclusion

In this work we investigated on evaluation methods for DCGANs. First, we trained a DCGANs model on CIFAR-10 dataset. The model successfully generated some images resembling the true CIFAR-10 images. Since the DCGANs model lacks a true objective function, it is hard to compare the performance of different classes and evaluate the performance for generated images compared with real images.

To evaluate the performance of GAN generator models, we investigated three different methods. The first method was a manual inspection experiment design to test if humans can recognize visually realistic images. Our volunteers were given 3 sets of 49 images, and each set of images contained approximately half true CIFAR-10 images and half fake images were generated by our DCGANs model. By analyzing how often our volunteers make wrong guesses, the performance of DCGANs model can be evaluated. However, this method was highly dependent on human judgement about visual quality of the images, which is unavoidably subjective. Despite the drawbacks, human evaluation is by far the simplest technique we investigated. In order to remove this impact, we introduced another evaluation method: Inception Score(IS). This method focuses on evaluating the image quality and image diversity of generated images. In other words, IS score reflects how well our DCGANs model can generate images that contain clear objects, and how diversified the images are. Based on the Inception Score, we explore another improved measurement, Frechet Inception Distance(FID). This method can measure the actual distance between Inception feature vectors of raw and generated images. Then we come up with our own unique way to measure how well our DCGANs model the underlying statistics of CIFAR-10 dataset in terms of each class: training a reverse-generator on DCGAN-generated images, and passing true images through the reverse-generator, subsequently reconstructing them with the generator. With this we were able to measure how well the generator could reconstruct each class. This gave us some insights into the DCGANs ability to represent the underlying statistics of the dataset.

To sum up, by calculating those three methods for GANs evaluation, we can know how our model performs and we can choose a particular evaluation method according to different needs. We summarize that our GANs generator performs well in generating fake images since they follow the real image distribution well due to a low FID score.

VI. Future Work

Based on what we've done so far, there are still some aspects we can further explore and improve in the future.

For the designed experiment, firstly, we can invite more volunteers to get involved in our experiment. A larger sample would provide more accurate and more convincing result. What's more, people participating in the experiment should have diverse backgrounds, since most of our friends who have done the experiment up to now have some knowledge about data science and neural networks. This might help them perform better than those who don't have related background knowledge. Thus, a group of people with different education background might provide more representative result. Also, for a fair result, we are considering using machine-built image annotators to judge the visual quality of samples since human performance is unstable and will improve to distinguish the fake images as they practice a lot.

For the DCGANs trainning process, we might consider about training DCGANs on different datasets to see if our evaluation metrics are consistent across datasets. We could have built different image-construction neural networks to compare IS and FID score against, to give a baseline for the efficacy of our DCGANs.

For DCGANs performance evaluation, we can explore more methods, like considering some qualitative GANs Generator evaluations as well, such as Nearest Neighbors. Nearest Neighbors is also a popular and helpful approach to measure how realistic the images are by selecting some real images from the domain and comparing them with most similar generated images their location.

Acknowledgment

We would like to thank Prof.Orchard for providing guidance and suggestions on our project's topic and confirming that our ideas bring some novelty. We also would like to thank all the volunteers who have spent their valuable time on our designed experiments.

References

[1] A. Randford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," abs/1511.06434, 2015.

[2] A. Borji, "Pros and Cons of GAN Evalutation Measures," arXiv: 1802.03446 [cs], Oct. 2018.

[3] H, Martin, et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." Advances in neural information processing systems. 2017.

[4] T.Salimans,I.Goodfellow, W. Zaremba,V. Cheung, A.Radford, X. Chen, "Improved techniques for training gans." Advances in Neural Information Processing Systems, 2234–2242,2016.

[5] Y. Sagawa & M.Hagiwara "Face Image Generation System using Attributes Information with DCGANs." Transactions Of Japan Society Of Kansei Engineering, 17(3), 337-345, 2018.