CSCE 479/879 Homework 3: Generative models

Shubham Bry
Shiva Paudel
Puranjit Singh
Kantilata Thapa

April 5, 2022

Abstract

Genrative modeling is an unsupervised learning task that involves automatically discovering and learning the regularities or patterns in input data in such a way that model can be used to genrate new examples that possibily could have been drawn from the original dataset. Generative models are powerful techniques that aim at learning the true data distributions from the training data to generate new data points with same variations. The main objective of this homework was to develop a generative model like Generative Adversarial Network (GAN) using CIFAR10 dataset. The main motivation was to develop a GAN on the CIFAR10 dataset and to evaluate its performance. The validation accuracy for the reconstructed images was calculated to be ... during the training phase. The FID score for the new instances generated was around 1.78.

1 Introduction

The main goal for this homework assignment 3 was to develop a generative model like a variational autoencoder (VAE) or a generative adversarial network (GAN) either using CelebA dataset or a dataset like CIFAR10. We chose CIFAR10 dataset to work on this problem because the training process was quite easier and faster in CIFAR10 dataset than that of CelebA dataset. Working with a familiarized dataset such as CIFAR10 helps us focus more on aspects such as architectural tuning and variations in the sets of hyper-parameters used in our homework. Also, this helped us to train the model in a lesser amount of time for the image generation process.

Specifically, we used a generative adversarial network (GAN) for this homework assignment. The GAN (Generative Adversarial Network) consists of two models, the generator model, and the discriminative model. A generator network takes a sample from random noise and generates sample of data and discriminator network decides whether the data is generated or taken from the real sample using a binary classification problem with the help of sigmoid function

that gives the output in a range between 0 and 1. Generative model analyzes the distribution of the data in such a way that after the training phase the probability of discriminator making a mistake maximizes and the discriminator on the other hand is based on a model that will estimate the probability that the sample is coming from the real data or the generator.

We used discriminator model which has 4 convolution layers with stride size of 2×2 to down sample the input image. For the generator architecture, it requires to transform a vector from the latent space with 100 dimensions to a 2D array with $32 \times 32 \times 3$ or 3,072 values. 48 nodes were used to represent a low resolution output image in the inital layer. The crucial operation for this architecture was up sampling of low resolution images which was achieved by sampling with Conv2D Transpose layer. The stride of size 2×2 was used after every Conv2D transpose layer. Similarly, kernel size of 4×4 was used to avoid a checkerboard pattern. Finally, the FID score of 1.78 was obtained to evaluate the model performance [1].

2 Problem Description

The major problem of this homework assignment is to build a generative model based on VAE (Variational Auto Encoder) or GANs (Generative Adversarial Network) to train and generate new samples from the CelebA image datasets or other datasets. We used CIFAR10 datasets and GAN model to address the specific problem for this homework assignment.

2.1 CIFAR10 datasets

CIFAR10 dataset consists of 60,000, 32×32-pixel color images of objects with 10 classes, 6000 images per class. The 10 classes present in the dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck [3]. These are exceedingly small images, much smaller than a typical photograph. Therefore, they can be developed and trained quickly, allowing focus to be put on the model architecture and image generation process itself.

2.2 Approaches

Generative Adversarial Network (GAN) is an approach for fitting convolution neural networks to generate images. Building a GAN to generate images requires two different models, one discriminator convolutional neural network model to classify whether a given image is real or another generator model to generate a fake image.

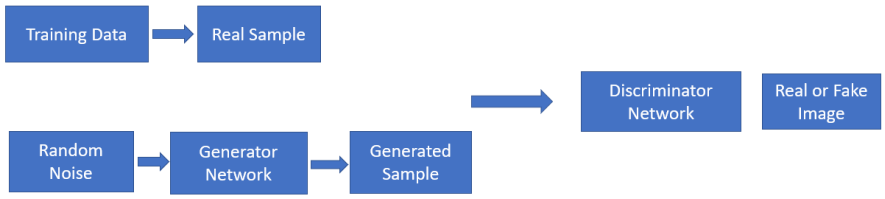

Figure 1: Workflow of GANs(Generative Adversarial Network

To generate an image generator model uses an inverse convolution layer to transform an input to a full two-dimensional image of pixel values. We defined a discriminator model, which takes an image with three color channels and 32×32 pixels as input and classifies the image whether it is real or fake(generated). Our discriminator model has four convolution layers with stride size of 2×2 to down sample the input image. For this model we used 'Adam' as optimizer. The architecture of fig 2 is our discriminator model.

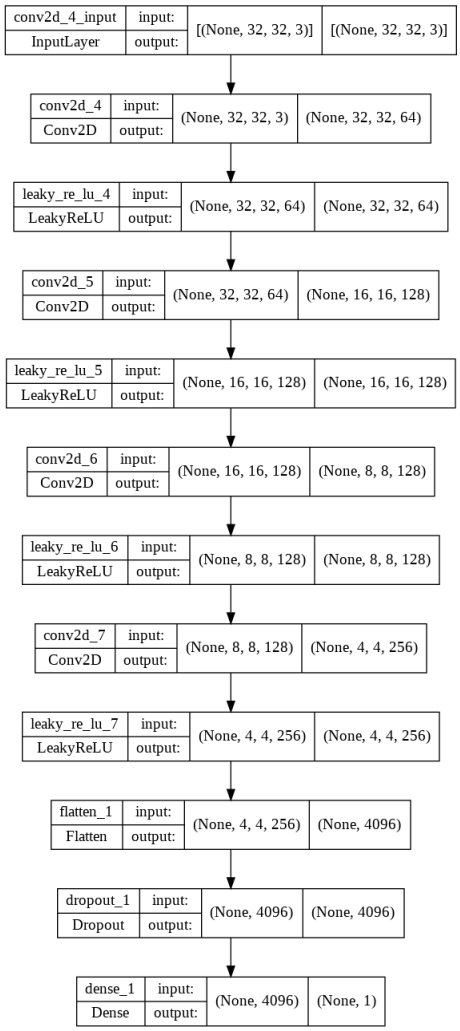

Figure 2: Discriminator Model Architecture

The generator architecture, fig 3 generates the fake image. This model takes latent space an arbitrarily defined vector space of Gussian-distributed values as input and a square color image as output. Developing a generator model requires that we transform a vector from the latent space with 100 dimensions to a 2D array with $32 \times 32 \times 3$, or 3,072 values [4]. We achieve this by deep convolutional generative adversarial networks. The initial layer in the generator model is a dense layer which has enough nodes to represent a low resolution version of the output image. We used a dense layer with 48 nodes to represent a low resolution output image. The crucial operation here is upsampling of this low resolution image. We achieve this by sampling with the Conv2DTranspose

layer. In our architecture we are using stride size of (2×2) to double after every Conv2DTranspose layer. It is also important to notice that we are using kernel size of (4×4) to avoid a checkerboard pattern. We repeated this layer unless we get an output image of size 32×32. The generator model weights are updated based on the performance of the discriminator model. The updates on the generator model are based on the classification results from the discriminator model, better the discriminator model performs higher will be the generator update. Finally, both the models were put together with a GAN model to fit both the models.

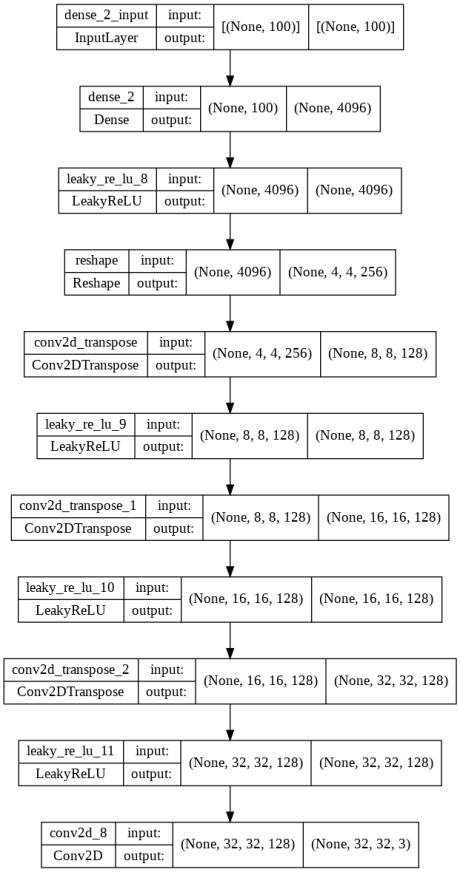

Figure 3: Generator Model Architecture

It was tricky to evaluate the performance of the GAN model as we cannot calculate error scores for generated images. To make the training process easier and effective, we calculated FID (Fréchet Inception Distance) scores on real and fake images. We programme the model to stop on training after our model performance i.e. FID score is similar for 5 epochs.

3 Experimental Setup

This section describes the process we used for training the GANs model so that the readers can replicate the implementation if needed.

3.1 Data Sources

To implement and train the GANs, we used CIFAR10 datasets [3]. This dataset was imported from keras API. The dataset was divided into five training batches and one test batch, with 10000 images each. The test batch contains exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Amongst them, the training batches contain exactly 5000 images from each class.

3.2 Pre-processing

For preprocessing of the datasets, keras provides access to the CIFAR10 dataset via cifar10.load_dataset() function. The pixel values of the images were scales from the range of unsigned integers in [0,255] to the normalized range of [-1,1].

3.3 Performance Measures

3.3.1 FID (Fréchet Inception Distance) Score

The FID score was used to evaluate the quality of images generated by generative adversarial networks, and lower scores have been shown to correlate well with higher quality images. To maintain consistency in the quality of the images that are generated, FID metrics were used. Lower scores indicate the two groups of images are more similar, or have more similar statistics, with a perfect score being 0.0 indicating that the two groups of images are identical. In other words, the similarity between real and generated images is close. FID compares the statistics of generated samples to real samples, instead of evaluating generated samples in a vacuum [5].

3.3.2 Generator Loss

While the generator is trained, it samples random noise and produces an output from that noise. The output then goes through the discriminator and gets classified as either "Real" or "Fake" based on the ability of the discriminator to tell one from the other. The generator loss is then calculated from the discriminator's classification – it gets rewarded if it successfully fools the discriminator and gets penalized otherwise.

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

The following equation is minimized to training the generator:

3.3.3 Discriminator Loss

While the discriminator is trained, it classifies both the real data and the fake data from the generator.

It penalizes itself for misclassifying a real instance as fake, or a fake instance (created by the generator) as real, by maximizing the function below.

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \left[logD\left(x^{(i)}\right) + \left(1 - D\left(G\left(z^i\right)\right)\right) \right] \tag{2}$$

$log(D(x))$ refers to the probability that the generator is rightly classifying the real image,

maximizing $log(1 - D(G(z)))$ would help it to correctly label the fake image that comes from the generator [2].

4 Experimental Results

As mentioned in Section 3, the model architectures were run for 200 epochs with early stopping callbacks and patience factor of 5. We can see from the results of FID values plotted in the following plot show that the values of FID was fluctuating around 2 after training model for 25 epcohs. We know that less is the FID value more is the simlarity between the original and genrated images. Observing values of FID around 2 in the training phase makes sense because we have 50000 images in our training dataset and only 5000 images per each class which is also a limitation in training these GAN better. Training GAN on CelebA dataset generates better results as compared to CIFAR10 dataset because there are approximately 200000 images present in the former.

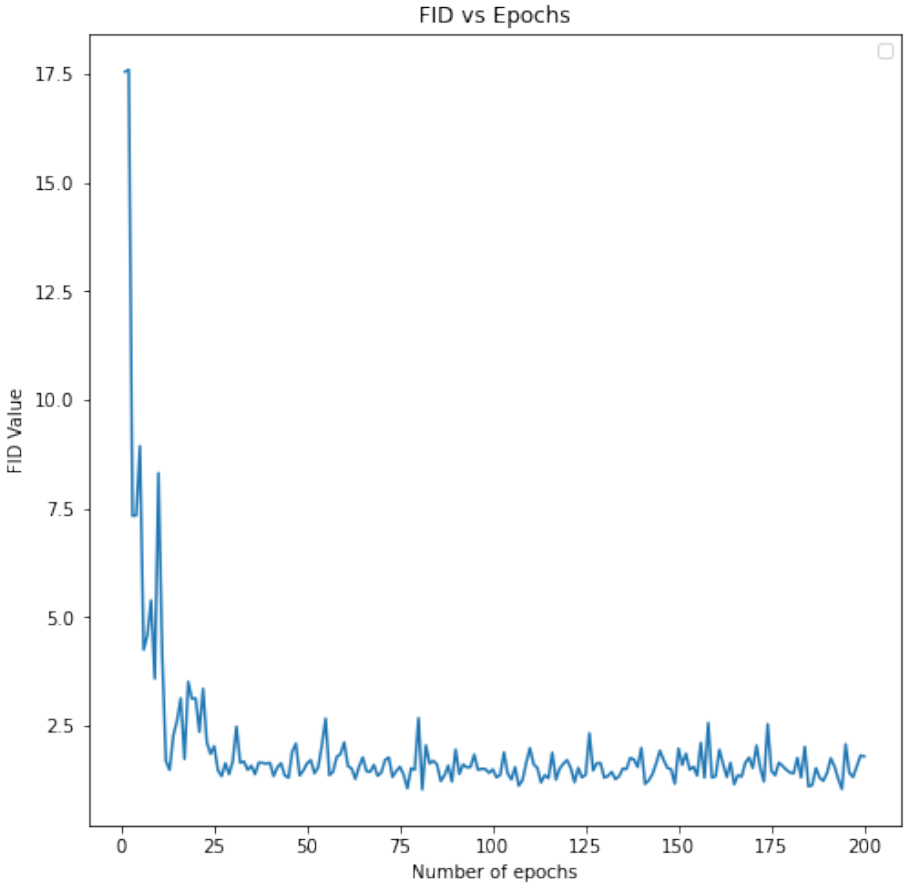

Figure 4: FID values vs No. of Epochs

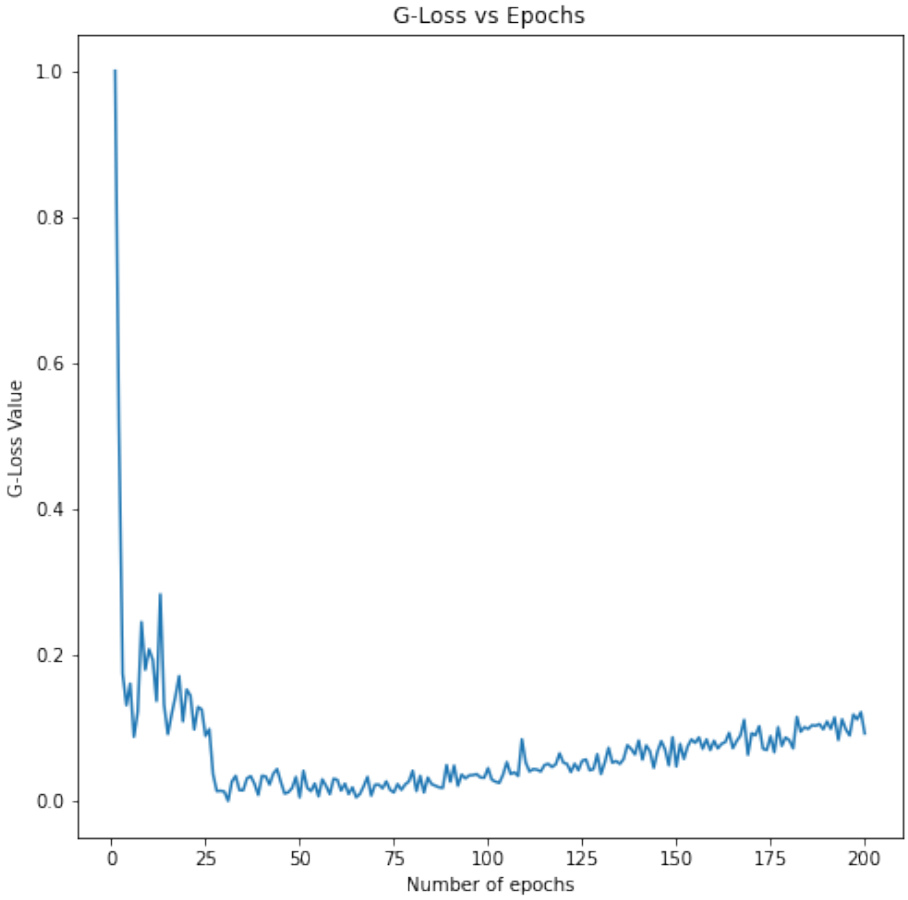

Figure 5: G loss values vs No. of Epochs

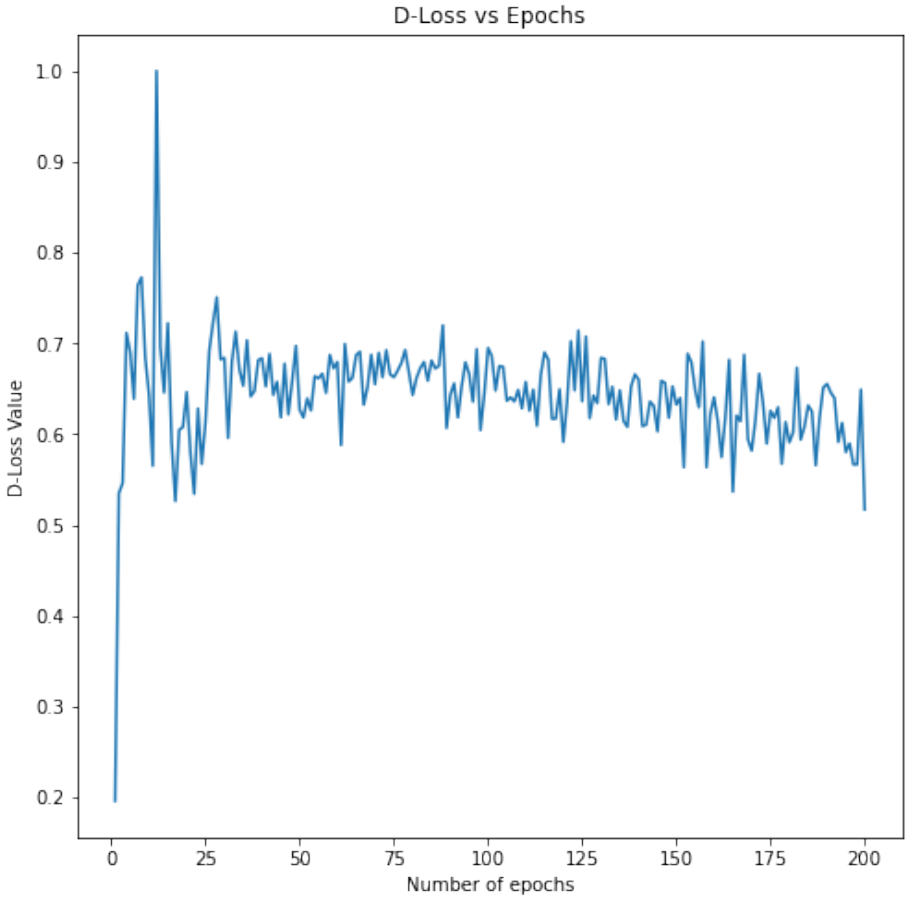

Figure 6: D loss values vs No. of Epochs

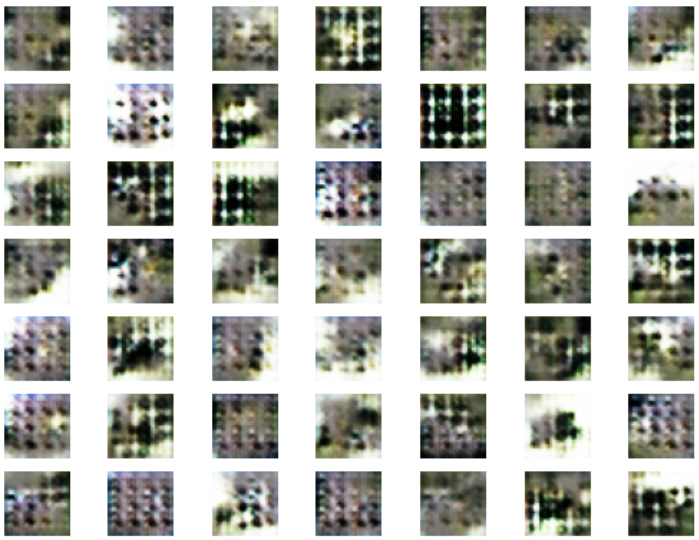

Figure 7: Generated Image on 10 Epochs

The result from generator model after 10 epochs are very low in quality but still some differences between background and foreground can be seen with a blog in the middle of each image.

Figure 8: Generated Image on 20 Epochs

Similarly, the images started to become little more clearer with increase in the number of epochs to 20 but still they lack clarity.

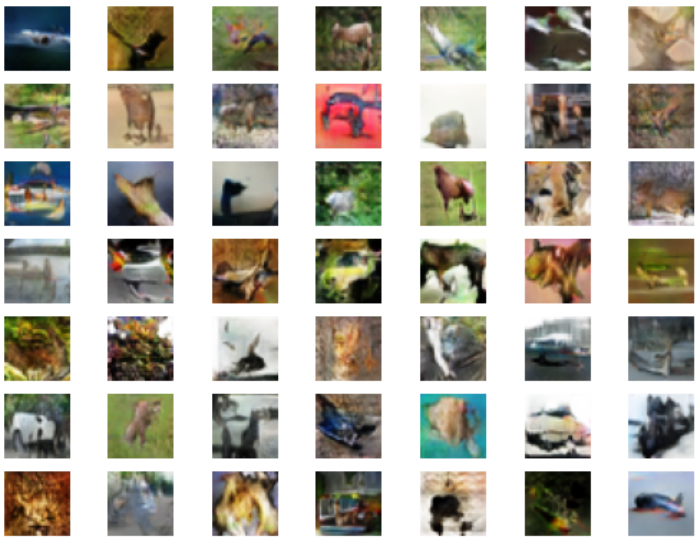

Figure 9: Generated Image on 200 Epochs

Finally at 200 epochs, we can see plausible photographs with blobs that look like birds, dogs, cats and horses. The objects look familiar to the objects in CIFAR10 dataset, but many of them does not belong to 10 specified classes of CIFAR10 datasets as shown in Figure 9.

5 Discussion

For this homework assignment, we developed a GAN model trained on CIFAR10 dataset. The rounded discriminator loss was around 0.88 with normalized generator loss of 0.924 after 2—epochs, implying that the GAN model was able to reconstruct the images mostly in the correct pixel value range. An important aspect we noticed about the developed model is the fact that there is little overfitting. The fact that there is no overfitting helped us with the decision of increasing the parameter space of both discriminator and generator by increasing the hidden layers. This gave a slightly better performance with a small increase in training time. This assignment helped to understand how to deal with complex architectures. It was tricky to evaluate the performance of the GAN model as we cannot calculate error scores for generated images. To make the training process easier and effective, we calculated FID (Fréchet Inception Distance) scores on real and fake images. We managed to get an FID score vary-

ing around 2. To speed up model training we used a GPU (Graphics Processing Units) setup.

6 Conclusions

For the learning problem of this homework assignment, we created a generative model based on GAN (Generative Adversarial Network) architecture using CIFAR10 dataset. To develop this model, hyper-parameters had to be trained from scratch as per the requirements. We measured the model's performance using the class competition and it got an FID score varying around 2. This homework assignment helped us get a good understanding of the methods to reduce computational complexities involved in dealing with deep architectures. As we did not observe any overfitting in the initial trial runs even with a lesser number of layers in the generator model, we increased the parameter space that enabled the model to give a better performance. Going ahead with this problem, future work could consist of developing a VAE (Variational Auto Encoder) and observing how that architecture compares with the GAN architecture we have already trained.

Table 1: Contributions by team member for this assignment.

Team Member	**Contribution**
Shubham Bery	Model implementation and result discussion
Shiva Paudel	Helping on model implementation and approach description
Puranjit Singh	Experimental Results Discussion
Kantilata Thapa	Helping on model implementation and introducing the problem and datasets

References

[1] How to Develop a GAN to Generate CIFAR10 Small Color Photographs. `https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-a-cifar-10-small-object-photographs` Accessed: 2019-07-01.

[2] How to Implement the Frechet Inception Distance (FID) for Evaluating GANs. `https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/`. Accessed: 2019-08-30.

[3] The CIFAR-10 and CIFAR-100 dataset. `https://www.cs.toronto.edu/~kriz/cifar.html`. Accessed: 2009-11-12.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *Advances in neural information processing systems*, 31, 2018.