Reproducibility Mini Project 4

Rahma Nouaji, Mohammad Ghavidel, Bita Farokhian

Reproducibility Summary

Scope of Reproducibility

The main claim of the original paper Goodfellow [2016] by Ian Goodfellow et al. titled "Generative Adversarial Networks" is that the proposed Generative Adversarial Network (GAN) framework is capable of generating high-quality synthetic samples that are indistinguishable from real samples. The authors claim that GANs can learn to generate data that is similar to real data by training a generator network to produce samples that are close to real data and training a discriminator network to differentiate between real and fake data. The generator and discriminator networks are trained in a two-player minimax game, where the generator tries to produce fake data that is similar to real data, while the discriminator tries to distinguish between real and fake data. Overall, the main goal of reproducing this paper would be to confirm whether the proposed GAN framework is indeed capable of generating high-quality synthetic data, and to investigate whether the findings reported in the original paper can be replicated.

Methodology

Our project aimed to replicate the experiments outlined in the seminal paper on Generative Adversarial Networks (GANs) by using open-source code from the authors Goodfellow [2016], the published paper Goodfellow et al. [2014], and code provided by Brownlee Brownlee [2019]. By exploring the widely recognized and promising GAN framework in deep learning, we sought to gain further insight into its potential applications. Our team successfully reproduced the results reported in the original paper for both the MNIST and CIFAR-10 datasets. In addition to replicating the experiments, we conducted an ablation study on the MNIST dataset to determine which components were essential for achieving a certain level of performance and which were not.However, due to limited computational resources and time constraints, we were only able to conduct the ablation study on the MNIST dataset.

Results

We replicated the process of generating images for two datasets - MNIST and CIPHAR 10 - using almost the same hyperparameters, except for the number of epochs, which was adjusted due to resource limitations. Although the generation process involves randomness, we made our best efforts to reproduce the results as accurately as possible. While the results presented in the original paper were better than our own, we were still able to generate images that were similar to those in the MNIST and CIPHAR 10 datasets. To evaluate the quality of the generated images, we relied on visual inspection to assess how closely they resembled the real images.

What was easy

Reproducing the results on the MNIST and CIPHAR 10 datasets were relatively easy for us. The code provided by Brownlee Brownlee [2019] was well-documented and easy to follow, which made it easy to replicate their experiments. Additionally, the authors provided detailed descriptions of their methodology in their paper Goodfellow et al. [2014], making it easy for us to understand and implement their approach. As a result, researchers who are interested in applying GANs to their own problems could easily use the code and methodology provided in this paper as a starting point for their own work. We were already familiar with working on the two datasets used in the paper, which helped us in conducting the experiments. We wanted also to reproduce the results for Toronto Faces Dataset and we found out that it is no longer available Kaggle [Accessed April 25, 2023].

What was difficult

The author's code Goodfellow [2016] was outdated and could not be run due to compatibility issues with current Python versions. Despite our efforts to make it work, we were unable to run it and had to find an alternative source of

Preprint. Under review.

inspiration. We used the code provided by Brownlee Brownlee [2019] for implementation, while still referring to the author's code for hyperparameter values. Due to the time-consuming nature of training GAN models, we purchased a GPU to speed up the process. The ablation study we conducted was even more time-consuming, with some experiments running for up to two hours. Since many hyperparameters were not explicitly mentioned in the original paper, we faced difficulties in determining their values. Therefore, we decided to use commonly-used hyperparameters for GAN models instead.

1 Introduction

Generative Adversarial Networks (GANs) is a type of neural network that can generate new data samples by learning the underlying distribution of the training data. The GAN architecture consists of two networks; a generator and a discriminator. The generator tries to generate realistic samples that can fool the discriminator, while the discriminator tries to distinguish between the real and generated samples. The training process involves mainly a minimax game between the generator and the discriminator, where the generator learns to generate realistic samples that can fool the discriminator, and the discriminator learns to distinguish between real and generated samples. This process continues until the generator produces samples that are indistinguishable from real ones. Since the introduction of GANs in 2014 by Ian Goodfellow et al. Goodfellow et al. [2014], many researchers have applied them to various applications, including image synthesis, data augmentation, and domain adaptation.

In this project, we aimed to reproduce the experiments described in the original GAN paper and conduct an ablation study on the MNIST dataset. Our goal was to gain a better understanding of the GAN framework and to evaluate the importance of various components of the model. The ability to reproduce and analyze existing results is a crucial aspect of scientific research, and it allows for more reliable evaluation of the proposed methods.

2 Scope of reproducibility

The scope of reproducibility for our work is to replicate and extend the experiments presented in the seminal paper on Generative Adversarial Networks (GANs) by Ian Goodfellow et al. Goodfellow et al. [2014]. The main contributions of the original paper are: The introduction of GANs, a novel approach to training generative models by using a discriminator to distinguish between generated and real data. The evaluation of GANs on the MNIST, CIPHAR 10 and Toronto Face Dataset(TFD) datasets, demonstrating superior performance compared to traditional generative models, for instance, the generative stochastic network (GSN) framework Bengio et al. [2014]. The generative adversarial network (GAN) framework has both advantages (e.g., no Markov chains needed, wide variety of functions can be incorporated) and disadvantages (e.g., need for synchronization between Generator and Discriminator) compared to previous modeling frameworks, with some advantages being primarily computational and GANs being able to represent sharp distributions.

In this report, we aim to reproduce the main claims from the original paper, including:

1. GANs can generate realistic samples that are difficult to distinguish from real data on the MNIST and CIPHAR 10 datasets.
2. Ablation studies can provide insight into which components of the GAN architecture are essential for achieving good performance.

3 Methodology

We reproduced the experiments from Generative Adversarial Network Goodfellow et al. [2014] for both MNIST and CIPHAR 10 datasets. We also conducted an ablation study to analyze the individual components of GANs model and determine their impact on the overall performance. We opted to vary the batch size, activation function, loss function, latent dimension input to the generator, and, finally omit the dropout layer in the discriminator. To fully complete our work, we used Colab Pro GPU.

3.1 Model descriptions

We used a simple GAN architecture consisting of a generator and a discriminator network. The generator network had one fully connected layer with 100 as input dimension similar to the latent dimension and 7x7x128 neurons, and two transpose convolutional layers with 128 filters of size 4x4,padding set to same and strides of 2x2, followed by a convolution layer with a 'sigmoid' activation function to produce the final output image. The discriminator network had

two convolutional layers with 64 filters of size 3x3 and strides of 2x2, followed by two dropout layers with a dropout rate of 0.4, and two fully connected layers with 1 neuron each. The total number of parameters in this model was approximately 1.4 million.

3.2 Datasets

The CIFAR-10 dataset consists of 60,000 color images of size 32×32 pixels, divided into 10 classes, with 6,000 images per class. The training set has 50,000 images, the validation set has 5,000 images, and the test set has 10,000 images. The images in the CIFAR-10 dataset have been preprocessed by subtracting the mean pixel value and dividing by the standard deviation across the entire dataset. Additionally, the images have been zero-padded with 4 pixels on each side. The CIFAR-10 dataset can be downloaded from Krizhevsky [2009].

The MNIST dataset consists of 70,000 grayscale images of size 28×28 pixels, divided into 10 classes, with 7,000 images per class. The classes are the digits from 0 to 9. The training set has 60,000 images, and the test set has 10,000 images. The images in the MNIST dataset have been preprocessed by normalizing them to have a mean pixel value of 0.1307 and a standard deviation of 0.3081. Additionally, the images have been centered and rescaled to have a range of values between -1 and 1. The MNIST dataset can be downloaded from LeCun et al. [1998].

3.3 Hyperparameters

According to the Paper Goodfellow et al. [2014] and the open source code Goodfellow [2016] batch size: 128, For the optimizer, they used SGD with a learning rate of 0.0002, a momentum of 0.5, and mini batches of size 128 for training their GAN model. The number of epochs is not specified in the paper, but an example implementation in the accompanying GitHub repository Goodfellow [2016] uses 50,000 training steps as the stopping criterion with RMSprop as an optimizer, Latent space dimensionality: 100 (we found out that at first, they used a Gaussian Noise with sigma 0.2, however, they changed it to random noise) Activation function for both generator and discriminator: LeakyReLU, Loss function for both generator and discriminator: Binary cross-entropy. These hyperparameters are not fixed, certainly, they could be changed. While the original GAN paper used the RMSProp optimizer, many more recent implementations of GANs have used the Adam optimizer instead, as it has been shown to be more effective in practice for a wide range of deep learning tasks, hence we opted for this optimizer. We trained the GAN with a batch size of 128 and 100 epochs but found it to be time-consuming and prone to memory issues. Hence, we switched to a batch size of 256 as there was no significant difference in performance and it was faster to train.

3.4 Experimental setup and code

In this investigation, we followed the description outlined in the previous section to set the parameters for our experiments. Specifically, we established a batch size of 128 and a latent dimension of 100 for the base model, which was trained without conducting an ablation study, over the course of 100 epochs. The learning rate for GANs model on MNIST was the same as the article set on 0.0002. To assess the performance of our model, we monitored key metrics, including the training Binary Cross entropy loss of both the discriminator and the generator, as well as accuracy, after completing the 100 epochs. Furthermore, we relied on the visual inception of seeing the generated images to evaluate the effectiveness of our proposed framework in terms of model performance. We wanted to use other metrics such as Inception Score (IS) or Frechet Inception Distance (FID). However, they were computationally expensive since they rely on the Inception v3 model for the evaluation.
Our colab notebook code is accessible from here Code for ML project group 7.

3.5 Computational requirements

For our experiments, we utilized Colab Pro and the TensorFlow framework for model implementation and training. Our budget was mainly allocated toward GPU hours on Colab Pro.

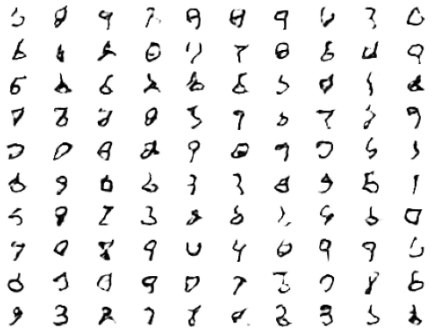
4 Results

In reproducing the GANs paper, we were able to generate images for the MNIST and CIPHAR 10 datasets using similar hyperparameters as the original paper, with the exception of the number of epochs due to resource constraints. Our results were similar to those reported in the original paper, although there were some minor differences due to the stochastic nature of the GANs algorithm. Increasing the number of epochs from up to 100 helped to improve the quality of the generated images. This can result in generated images that are closer to the real values, making it more difficult for the discriminator network to distinguish between real and synthetic data.

4.1 Results reproducing original paper

4.1.1 Model on MNIST dataset

As previously mentioned, we implemented the GANs model on the MNIST dataset using a batch size of 256 instead of 128 to expedite the process, as it was observed that the difference in results between the two batch sizes was negligible. Initially, we attempted to run the model for 20 epochs, but the output was unsatisfactory, as certain characters were indistinguishable. However, upon increasing the epoch count to 100, we observed significant improvements in the model's output, which was consistent with the findings of the original paper. Specifically, in Figure 1b, the image on the right displays the output generated after 100 epochs, while the image on the left demonstrates the inferior output generated after only 20 epochs. Overall, these results underscore the importance of conducting thorough experimentation and implementing appropriate training techniques to achieve optimal results in GANs modeling.



(a) output of the model after 20 epochs

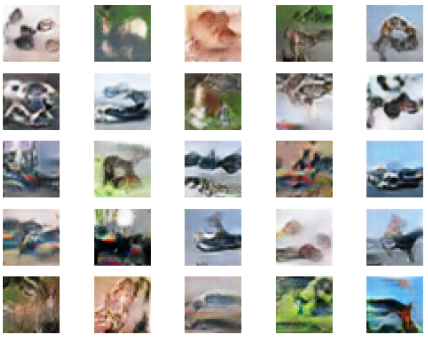
(b) output of the model after 100 epochs

Figure 1: Results of the GAN model for MNIST dataset

4.1.2 CIPHAR 10 dataset

As previously mentioned, we utilized the GANs model on the CIPHAR10 dataset and set the batch size to 256. Through experimentation, we determined that the choice of batch size and epoch count played crucial roles in the training of the model. Figure 2 displays the output of our experiments, where the image on the right was generated after 80 epochs and the image on the left was generated after only 20 epochs. It is evident from the figures that the output generated by the model after 80 epochs is more coherent and recognizable, while the output generated after 20 epochs is blurred and not well-trained. Nevertheless, except for the number of epochs which has a computational cost associated with its increase, the results we obtained using the GANs model on the CIPHAR10 dataset were consistent with those presented in the original paper. Thus, the selection of optimal parameters, such as batch size and number of epochs, is critical in producing effective and efficient models.

(a) output of the model after 20 epochs



(b) output of the model after 80 epochs

Figure 2: Results of the GAN model for CIPHAR dataset

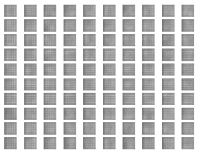
4.2 Results beyond original paper

In this section, we performed mainly an ablation study:

- We opted to drop the dropout layer in the discriminator network in order to see how the model would perform 3b.
- We went further to explore what happens if we change the binary cross entropy (BCE) loss function for the Discriminator model and GANs model to Mean Squared Error(MSE) 3c.
- We also explored the performance in case we change the activation function 3a.
- We made a grid search over the mini-batch sizes 4.
- We explored different latent dimensions for the generator (we did not put the results for this part because of space limitation constraints but we will discuss the results below).

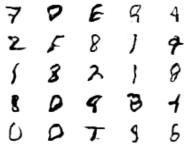
(a) Changing activation function to Sigmoid for all NN layers.

(b) Deleting the discriminator's dropout layer

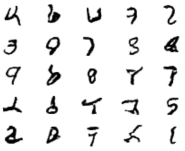


(c) Changing the BCE to MSE

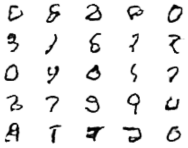
Figure 3: Ablation study conducted on GANS Model and MNIST Dataset(Training for 100 epochs)



(a) Image generation with Batch size 64



(b) Image generation with Batch size 128



(c) Image generation with Batch size 256

Figure 4: Grid search over the batch sizes (run for 50 epochs)

5 Discussion

5.1 Reproducing paper results:

In this study, we reproduced GANs model results Goodfellow [2016] by implementing the GANs model on two different datasets including MNIST and CIPHAR10. we highlighted the importance of selecting optimal hyperparameters, such as batch size and epoch counts. In the case of MNIST, we chose a batch size of 256 and observed that increasing the number of epochs resulted in significant improvements in the model's output with a learning rate of 0.0002, as we compared results for two 20 and 100 epochs; the results are consistent with the original paper but not exactly same due to computational limitation. The same trend was observed for CIPHAR10, where we increased the epoch count from 20 to 80, resulting in more recognizable outputs. It is noteworthy to mention that experimentation and appropriate training techniques are essential to achieve optimal results in GANs modeling.

5.2 Ablation study:

- As it is displayed in 3b, deleting the dropout layer from the discriminator led to poor performance. This could be explained by the fact that the discriminator can easily memorize the training data and fail to generalize to new data since dropout is a regularization technique that prevents overfitting. Therefore, it is essential to use dropout in the discriminator.

- Changing the BCE loss to MSE loss also led to poor performance which could be noticed from 3c, since the model completely fails to generate the images. This could be explained by the fact that MSE is used for regression tasks and we are in a classification task that is suitable for BCE loss. Therefore, it is essential to choose the appropriate loss function that is suitable for the task at hand.
- By changing the activation function from LeakyReLU to Sigmoid, the generator and discriminator networks are unable to learn the underlying distribution of the MNIST dataset and, as a result, fail to generate realistic digits, as you can see in 3a, it is not even recognizing the patterns of the digits. This can be attributed to the difference in their behavior. Sigmoid is a non-linear function that suffers from the vanishing gradient problem which causes the gradients to become very small, making the learning process too slow. On the other hand, LeakyReLU prevents the saturation problem. This leads to faster learning and better performance.
- For the batch size grid search 4, we ran it only for 50 epochs each because of computation constraints. We concluded that for batch sizes 256 and 128 the performance was close from a visual inception since we are comparing the results based on the generated images.
- We have also explored different latent dimensions from 20, 50 to 100. When reducing the latent dimension from 100 to 20, the generator network is unable to capture the complexity of the data, resulting in poorly generated images that are only recognizable as the digits 6, 9, and 0. This highlights the importance of using an appropriate latent dimension when training GANs on complex datasets.

5.3 What was easy

In this study, there were some points that helped us to proceed and made replication of results easier for us. The key points are as follows:

- We found it relatively straightforward to replicate the results on the MNIST and CIPHAR 10 datasets using the well-documented and easily understandable code provided by Brownlee [2019].
- The methodology described in the paper by Goodfellow [2016], which the authors referenced, was also helpful in implementing their approach. This makes it a convenient starting point for researchers interested in applying GANs to their own problems.
- Our familiarity with working on the MNIST and CIPHAR 10 datasets facilitated the replication of the results in our experiments. However, we encountered difficulties in reproducing results on the Toronto Faces Dataset, as it is no longer accessible on Kaggle as of April 25, 2023.

5.4 What was difficult

The original code provided by Goodfeli Goodfellow [2016] proved to be incompatible with current Python versions, necessitating our attempts to resolve compatibility issues, which unfortunately proved unsuccessful. Consequently, we had to seek alternative sources for implementing the proposed framework. After thorough consideration, we opted to utilize the code provided by Brownlee Brownlee [2019], which offered a more compatible and functional solution for our implementation needs. However, as the original code was still referenced for hyperparameter values, we encountered challenges in determining the appropriate values due to the lack of explicit guidance in the original paper.

To expedite the time-consuming process of training GAN models, we made the decision to purchase a GPU, which significantly improved the efficiency of our experiments. Nevertheless, the ablation study we conducted posed additional challenges, as it involved running experiments for extended periods of time, with some lasting up to two hours. This required careful planning and resource allocation to ensure accurate and reliable results.

One limitation we encountered was the lack of explicit hyperparameter values mentioned in the original paper, which required us to make decisions based on commonly-used hyperparameters for GAN models. Despite these challenges, we made diligent efforts to ensure the validity and reliability of our experimental results.

5.5 Communication with original authors

Not applicable

References

Yoshua Bengio, Eric Thibodeau-Laufer, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, pages 226–234. PMLR, 2014.

Jason Brownlee. How to develop a generative adversarial network for an mnist handwritten digits from scratch in keras. *Machine Learning Mastery*, May 2019. URL https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-kera

Ian Goodfellow. Adversarial examples in the physical world. https://github.com/goodfeli/adversarial, 2016. Accessed: 2023-04-23.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Kaggle. Kaggle: Your home for data science. https://www.kaggle.com/, Accessed April 25, 2023.

Alex Krizhevsky. Cifar-10: Canadian institute for advanced research. https://www.cs.toronto.edu/~kriz/cifar.html, 2009.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 1998.