Generative Modeling Using a DC-GAN

Timothy Taylor, Areen Lu
University of California, San Diego
tvtaylor@ucsd.edu, arlu@ucsd.edu

Abstract

Generative modeling has taken the world by storm, and shown amazing capabilities, particularly in image generation. In this paper, we explore the capabilities of Generative Adversarial Networks to produce realistic images. We test a DC-GAN model on two different datasets: the MNIST dataset which contains 10 classes, 28x28 grayscale images of hand drawn digits with labels 0 to 9; and the CIFAR-10 dataset, a dataset of 32x32 colored images with a diverse collection of ten different class labels including horses, cats, automobiles, planes, etc. We explored the effects of different parameters on the DC-GAN's performance on both datasets, accessing the quality of generated images through both visual inspection and quantitative measures, utilizing the inception score to provide a standardized evaluation. Additionally, we explore how different latent space dimensionalities affect the performance of the DC-GAN, applying the same evaluation methods. By systematically analyzing these factors, we aim to gain insights into the optimization of DC-GANs for improved image generation across different types of datasets. We ultimately find that the dimensionality of the latent space and training resources are the biggest factors determining the quality of the generated images, proving image generation to be a task that is unavoidably resource-demanding.

0 Group Members

Areen Lu: Helped with network architecture by testing various numbers of convolutional layers, kernel sizes, types of pooling layers, presence of dropout layers, numbers of neurons in the final linear layer, and various activation functions, and optimized the learning rates and optimization algorithm for peak model accuracy

Timothy Taylor: Structured network architecture, optimized network architecture by testing various numbers of convolutional layers, kernel sizes, types of pooling layers, presence of dropout layers, numbers of neurons in the final linear layer, and various activation functions, and optimized the learning rates and optimization algorithm for peak model accuracy

1 Introduction

In this report, we explore and summarize our findings about designing the architecture of the discriminator and generator in a Generative Adversarial Network (DC-GAN) used for image generation. Our primary objective is to investigate and summarize the design and performance of the DC-GAN's architecture, particularly the structures of the discriminator and generator, and how these influence the overall image generation process. The DC-GAN operates by pitting a generative network against a discriminative network. The discriminative network within the DC-GAN is a convolutional neural network (CNN) that takes in input images to classify them as real or fake. And in doing so, it learns the differences between real images from the dataset and fake images created by the generative network. The generative network is another CNN that is used to create images from random noise inputs. Its goal is to create images that are indistinguishable from the real ones. By using them against each other in training, the generative capabilities of the generator are maximized as it learns to "fool" the discriminator.

We train the model to generate images from both the MNIST and CIFAR-10 datasets. To evaluate the performance of our DC-GAN, we use the inception score, a widely accepted metric for assessing the quality of generated images, calculated using the InceptionV3 model from Keras. This score measures both the diversity and realism of the generated images by examining the distribution of predicted labels. We also conduct human inspection of the generated images to provide a subjective assessment of their visual fidelity and coherence.

We ultimately find that the dimensionality of the latent space affects the quality of the generated images the most, with a higher dimensionality leading to better results at the cost of runtime. While the DC-GAN struggles to generate images similar to those of the CIFAR-10 dataset, it has amazing potential for the MNIST dataset, producing some images almost indistinguishable as a fake by the human eye.

2 Methods

2.1 Problem motivation and description

Generative Adversarial Networks (GANs) have revolutionized the field of image generation and because of this, the Deep Convolutional GAN (DC-GAN) has gained popularity due to its ability to generate high-quality images. However, the performance of DC-GANs can vary significantly depending on the complexity and nature of the dataset they are trained on. Hence, we test its capabilities on two very different sets of images.

The MNIST dataset is the more simplistic set as it has less distinct categories, and limited variation in images since all classes are ultimately handwritten digits that are grayscaled and drawn in relatively the same size. Due to its simplicity, it serves as an ideal starting point for us with the DC-GAN model. In comparison, the CIFAR-10 is the more challenging and complex dataset as it has highly distinct class labels, with a lot more variation in the images as the classes include both vehicles, animals, and color. CIFAR-10's diversity introduces significant challenges for the DC-GAN as it must learn to generate images that capture the wide range of visual features of the dataset while maintaining the distinct class characteristics.

With the contrasting nature of both datasets, we are able to establish a baseline understanding of DC-GAN with the MNIST dataset while CIFAR-10 allows us to push the limits of the model. As a result, this provides us a comprehensive evaluation of the DC-GAN as it allows us to observe how the model adapts to different complexity levels and variations in the data. This will also showcase the strengths and weaknesses of architecture, providing insights into possible areas of improvement and optimization.

In conclusion, by seeing the performance of DC-GAN model on both datasets, MNIST and CIFAR-10, we desire to gain a deeper understanding of how the factors influence generative modeling success by assessing the impact of hyperparameters such as the convolutional layers in the model, stride, and kernel size. .

2.2 Data processing and selection

We chose to work with two datasets: the MNIST handwritten digits dataset and the CIFAR-10 dataset, each presenting different challenges for the DC-GAN model . The MNIST dataset consists of 60,000 training images, and 10,000 test images of handwritten digits in a 32 x 32 grayscale image with an appropriate label in the range 0 to 9. The CIFAR-10 dataset consists of 50,000 training images, and 10,000 test images that display a 32 x 32 three-color-channel photo of one of ten classes: "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", and "truck". No image in the CIFAR-10 dataset has multiple labels.

Our rationale for selecting these datasets lies in their contrasting characteristics. MNIST has simpler images that are easier to learn. This acts as a controlled environment as it focuses on the basic shapes and structures. While the high variety of the images in the CIFAR-10 dataset is the more challenging dataset for a generative model due to the broader spectrum of object types and visual styles.

2.3 Model architecture

We based our model on the DC-GAN architecture outlined in Jason Brownlee's "How to Develop a GAN to Generate CIFAR10 Small Color Photographs". We used TensorFlow to implement the DC-GAN through two parts: the generator, and the discriminator.

For the discriminator, we designed a convolutional neural network that produces a single scalar output that determines the binary label of "real" or "fake" to images. The model contains 4 convolutional layers, each with a stride of 2 and kernel size of 4. We use Leaky ReLU as the activation function for each layer in order to introduce non-linearity.

The generator contains 2 transposed convolutional layers, each with a stride of 2 and kernel size of 2. It also contains one convolutional layer with a stride of 3 and kernel size of 1 to further refine the generated images. Similar to the discriminator, we again use Leaky ReLU as the activation function for each layer.

We ultimately found that the presence of max pooling layers, as well as changing the stride and kernel size of the convolutional layers had minimal impact on the performance of the DC-GAN. What mattered the most in our testing was the dimensionality of the latent space; we found that a higher dimensionality allowed for better results at the cost of runtime. This conclusion led us to our choice of 100 for the dimension of the latent space.

2.4 Training procedure

A DC-GAN is trained by utilizing both the generator and the discriminator. The generator is used to create a batch of fake images from some input vectors in its latent space, and the discriminator is then trained to classify the images as either real or fake, real from the dataset or fake as in generated by the generator. The DC-GAN uses the loss of the discriminator to improve the performance of the generator: the more loss the discriminator has, then the better our generator is at "fooling" the discriminator. We use this training procedure to create and evaluate models with varying architectures and hyper-parameters, and we evaluate each iteration of our model. During this training, we used a learning rate of 0.0002, and ADAM as our optimizer as it adjusts the learning rate dynamically for each parameter based on the estimates of the first and second moments of the gradients.

For better results we trained the DC-GAN for 110 epochs on the CIFAR-10 dataset, and 70 epochs on the MNIST dataset. The CIFAR-10 had more training due to the level of complexity the dataset naturally has. However, due to the extensive cost of runtime and computational power needed, we used the results of training on 10 epochs to compare the effects of different dimensionalities for the latent space and hyperparameters.

2.5 Evaluation metrics

To evaluate the performance of our model, we use the inception score and visual inspection by humans. The inception score is a commonly used algorithm to evaluate the performance of generative models, and it gives us a quantitative measurement of our model's performance. We use the Inception V3 model from keras to implement the inception score. A higher inception score indicates a higher variety and distinctness in our generated images (i.e. there are different images, and the images fall into distinct classes as we desire). We calculate the inception score across 100 generated images, comparing it to the inception score we obtain from 100 plots of random noise, or 100 of the real MNIST images.

In addition to evaluation with the inception score, we also do human visual inspection on the generated images. This allows us to provide further and human opinionated insights to the overall quality and realisticness of the images. By evaluating the image details such as image clarity, structure and similarity to the original dataset, we then are able to provide a more thorough understanding of the model's performance

By using these two evaluation methods, it allows us to iteratively refine our model architecture, adjust hyperparameters, and optimize training strategies to achieve better image generation results across both MNIST and CIFAR-10 datasets.

3 Experiment and Results

3.1 Optimizing discriminator hyperparameters

To optimize the architecture of the discriminator, we first tested its performance with kernel sizes of 2, 3, and 4. As in *Figures 1-3*, there is minimal visual difference between the produced images, and we see from *Table 1* that the inception score is essentially equivalent for all generated images. In fact, the inception scores for the generated images are almost as low as that of random noise, but visual inspection shows that our generated images are far from random. They have definitely taken on some patterns of the CIFAR-10 dataset; for example, *Figure 1* looks to be some sort of animal in a field, with a clear shadow and grassy background. Moreover, even the inception score for the real images is only 2.2631, which is quite low already. The low inception scores are likely due to the fact that the MNIST dataset has low variety; each image is grayscale, and of a digit that is roughly the same size and in the center of the image. Hence, we rely more on visual inspection to guide our results on the images generated from the MNIST dataset. Upon visual inspection, our results do not appear to significantly differ in quality regarding kernel sizes.

Figure 1: Discriminator Kernel Size of 2 on CIFAR-10

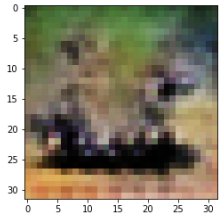

Figure 2: Discriminator Kernel Size of 3 on CIFAR-10

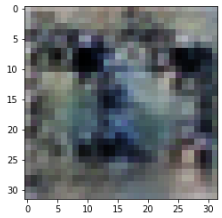

Figure 3: Discriminator Kernel Size of 3 on CIFAR-10

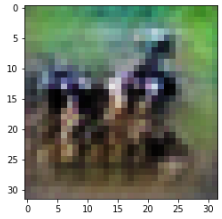

Table 1: The Effect of Kernel Size of the Discriminator on Inception Score of Generated Images

Kernel Size	**Inception Score**
Ground Truth on Kernel Size 4	2.2632
Random Noise on Kernel Size 4	1.0000
2	1.0004
3	1.0002
4	1.0008

We also tested the effectiveness of adding max pooling layers, giving us the results seen in *Figures 4-5*, and *Table 2*. Again, there is no significant difference visually, nor with regard to the inception scores. Given these results, we decided it best to keep max pooling layers absent from the discriminator for the sake of runtime.

Figure 4: Max Pooling Absent in Discriminator on CIFAR-10

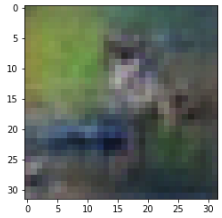

Figure 5: Max Pooling Present in Discriminator on CIFAR-10

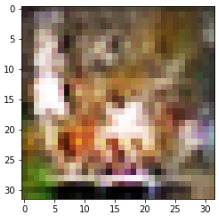

Table 2: The Effect of Max Pooling in the Discriminator on Inception Score of Generated Images on CIFAR-10

Max Pooling Present/Absent	**Inception Score**
Present	1.0004
Absent	1.0005

Lastly, we test the effectiveness of different stride lengths, giving us the results seen in *Figures 5-7*, and *Table 3*. The inception scores once again hold little difference, but the differences are of a larger magnitude than before. From the inception scores, we see that a stride length of 2 is the best, followed by a stride length of 4. Visually, these results are reaffirmed. *Figure 5* is clearly depicting a one, while *Figure 6* depicts a more blurry figure whose digit is harder to recognize. As seen in *Figure 7*, a stride length of 4 generates an image that is more visually clear than that of a stride length of 3, but it is still not as clear as in *Figure 5*. The digit in *Figure 7* looks closest to a 3, but also similar to an 8. Hence, we conclude that a stride length of 2 is best for the discriminator, and we cement it into the discriminator architecture.

Figure 5: Discriminator Stride Length of 2 on MNIST

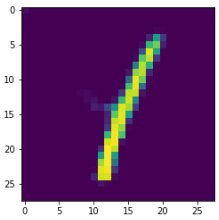

Figure 6: Discriminator Stride Length of 3 on MNIST

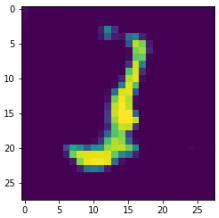

Figure 7: Discriminator Stride Length of 4 on MNIST

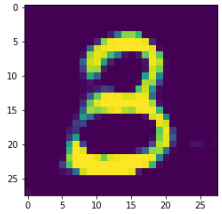

Table 3: The Effect of Stride Length of the Discriminator on Inception Score of Generated Images on MNIST

Stride Length	**Inception Score**
2	1.0081
3	1.0009
4	1.0032

3.2 Optimizing generator architecture

In addition to optimizing the architecture of the discriminator, we also seek to optimize the architecture of the generator. We tested kernel sizes of 2, 3, and 4 in the transposed convolutional and convolutional layers in the generator network. As seen in *Table 4*, there was no significant difference between the inception scores of the generated images; they differed by only a magnitude of tenths of thousandths. Visually, all the images are quite clear as well, as we observe in *Figures 8-10.* From our results, it seems that the architecture and hyperparameters of the discriminator are more important than those of the generator. Moreover, it seems that the performance of the DC-GAN as a whole is not too reliant on hyper-parameters so long as the structure of the networks remains the same, since only the stride length of the discriminator has so far had any noticeable effect on our generator's performance. We thus conclude that in terms of architecture, changing the stride length of the discriminator yields the most tangible results.

Figure 8: Generator Kernel Size of 2 on MNIST

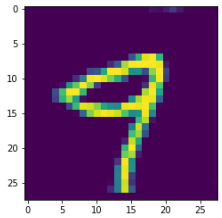

Figure 9: Generator Kernel Size of 3 on MNIST

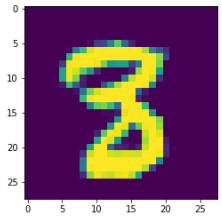

Figure 10: Generator Kernel Size of 4 on MNIST

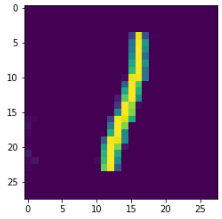

Table 4: The Effect of Kernel Size of the Generator on Inception Score of Generated Images on MNIST

Kernel Size	**Inception Score**
2	1.0004
3	1.0002
4	1.0008

3.3 Optimizing dimensionality of the latent space

In addition to the hyperparameters of the discriminator and generator, we also tested the influence of the dimensionality of the latent space that the generator uses as a basis for creating images. We tested latent dimensions of 25, 50, and 100. As seen in *Table 5*, increasing the dimensionality led to better performance as evaluated by the inception scores of the generated images. Moreover, we see this directly in *Figures 11-13*, which display examples of images generated from each different dimensionality. In *Figure 11* the image is closer to a blob than any real di git, whereas it closely resembles an 8 in *figure 12*. The best generated image is seen in *Figure 13*, which is a result of using a dimensionality of 100. We hence conclude that a higher dimensionality will lead to better image generation, though at the cost of performance since a higher dimensionality increases the complexity of the generator model.

Figure 11: Latent Space Dim of 25 on MNIST

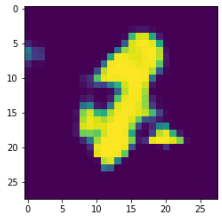

Figure 12: Latent Space Dim of 50 on MNIST

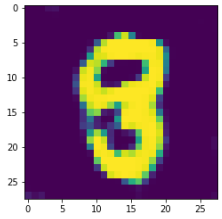

Figure 13: Latent Space Dim of 100 on MNIST

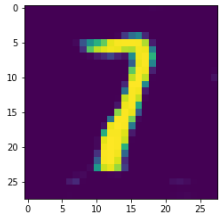

Table 5: The Effect of Latent Space Dimensionality on the Inception Score of Generated Images on MNIST

Latent Space Dim	**Inception Score**
25	1.0003
50	1.0025
100	1.0093

3.4 Results by Epoch

Due to resource limitations regarding both computing power and time, the longest we were able to train the DC-GAN was for 70 epochs on the MNIST dataset, and 110 epochs on the CIFAR-10 dataset. We demonstrate the model's marked improvement in image generation over these epochs in *Figures 14-15*. From epoch 10 to 70 on the MNIST dataset, the DC-GAN's generated image goes from a blob to a distinct 9 that is indistinguishable from real digits in the MNIST dataset. It's inception score also increases as seen in *Table 6*.

These marked improvements over time are also present in *Figures 16-17* which show the DC-GAN's improvement in generating images in the CIFAR-10 dataset. The first image generated in epoch 10 is unrecognizable as anything, but the image generated in epoch 110 is clearly some sort of animal, perhaps a bird on a branch. As the epochs increase, there is marked improvement in the DC-GAN's image generation, implying that time and energy spent on training may well be the most important resources in determining a DC-GAN's performance.

Figure 14: Generated Image from Epoch 10 on MNIST

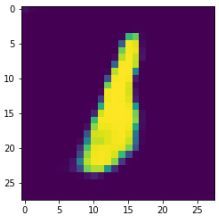

Figure 15: Generated Image from Epoch 70 on MNIST

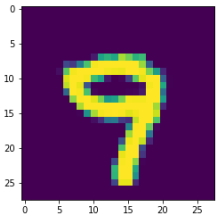

Figure 16: Generated Image from Epoch 10 on CIFAR-10

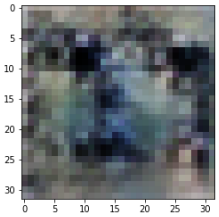

Figure 17: Generated Image from Epoch 110 on CIFAR-10

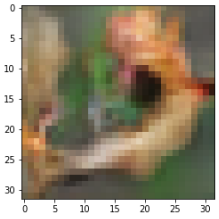

Table 6: Generated Image Improvement over Epochs

Epoch/Dataset	**Inception Score**
10/MNIST	1.0021

70/MNIST	1.0075
10/CIFAR-10	1.0031
110/CIFAR-10	1.0092

3.5 Limitations

Within this project, it is important to recognize some of the limitations that came with the DC-GAN architecture. Training GANs are computationally intensive due to the adversarial training where the discriminator and generator continuously improve as a response to each other. With our available hardware, when scaling up the complexity and duration of training that is required for optimal GAN performance, we had bottlenecks when scaling up the complexity and duration of the training. In order to counter this, we created a balance between the model complexity and training duration as stated above with the epochs. Along with the hardware constraints, time is a critical factor with the training GANs to convergence. Due to this, we had to make strategic decisions regarding the model run times so that we can perform multiple changes to the model, iterate model improvements, and thoroughly evaluate the outcomes of our results.

Conclusion

In this project, we explored the capabilities and limitations of DC-GAN for generating realistic images from the two datasets: MNIST and CIFAR-10. We evaluated how various architectural choices, hyperparameters, and the dimensionality of the latent space affect the quality of the generated images. Our findings showed the MNIST dataset with the DC-GAN generates visually convincing images relatively quickly. The CIFAR-10 dataset, however, posed a challenge for the DC-GAN, highlighting the limitations and areas for improvement in generative modeling. Overall, this showed the importance of balancing model complexity, training duration, and computational resources to achieve high-quality image generation using DC-GANs. While the DC-GAN architecture showed promising results for the simpler MNIST dataset, it faced challenges with the more complex CIFAR-10 dataset, suggesting that further optimization and more powerful computational resources are necessary for generating high-quality images from diverse datasets. Future work could explore advanced GAN architectures to see their performances on more complex datasets like the CIFAR-10. By continuing to push the boundaries of GAN research, we can further unlock the potential of generative modeling for a wide range of applications in computer vision and beyond.

References

Brownlee, J. (2020). *How to Develop a GAN to Generate CIFAR10 Small Color Photographs*. Retrieved from https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-a-cifar-10-small-object-photographs-from-scratch/

Canadian Institute for Advanced Research. (2009). CIFAR-10 (Canadian Institute for Advanced Research). Retrieved from https://www.cs.toronto.edu/~kriz/cifar.html

LeCun, Y., Cortes, C., & Burges, C. J. C. (2010). MNIST handwritten digit database. Retrieved from http://yann.lecun.com/exdb/mnist/