Conditional Generative Adversarial Networks (cGANs): A Re-implementation of cGANs for ECE 50024 Project

Sai Soham Ramagiri, ECE, MS [1]

Abstract

Conditional Generative Adversarial Networks are a type of Generative Adversarial Network that can generate specific samples based on a set of conditions. The cGAN's generator network produces a numerical image identical to a real image, taking a random noise vector and a conditioning label as its input. The discriminator network is trained to distinguish between realistic and false images. The suggested method produced satisfactory outcomes for the MNIST dataset and demonstrated the potential of cGANs for image synthesis applications.

1. Introduction

Although Generative Adversarial Networks (GANs) have become a key tool in recent times with a wide range of applications, and have seen remarkable success in producing high-quality synthetic data samples by learning the underlying distribution of the training data, they are considered challenging to train and might generate poor samples, especially when the data distribution is intricate.

Conditional Generative Adversarial Networks (cGANs), which enable the generator to learn the mapping from an additional input condition to the output data distribution, were hence developed by Mirza and Osindero as a solution to some of these problems in Generative Modeling. These cGANs can produce specific samples conditioned on a label or class and have made substantial progress in many disciplines, including signal processing, computer vision, and natural language processing. ”The cGAN model can be used to generate new images that are highly realistic and semantically meaningful” (Mirza & Osindero, 2014)

The authors provide a cGAN architecture for training on MNIST data, that learns a mapping from a conditional input to the output image and is based on a two-player minimax game. The discriminator network tries to tell real images from false ones. The generator network works to create images that the discriminator may use to distinguish between fake and real ones.

This project's primary goals are to examine how well cGANs can produce context-conditioned images on MNIST data, and find how different hyperparameters affect the output's quality. Specifically, the impact of the conditioning variable, the generator and discriminator architectures, and the loss functions on the quality of the generated images is being investigated. This can help us design better cGAN networks with improved model performance. The study also examines the impact of several hyperparameters, such as the learning rate and the number of epochs, on the functionality of the cGAN architecture. This helps in the optimization of the training process and ensures the generated images are of high quality. Therefore, addressing these sub-problems is critical to advancing our understanding of cGANs and improving their practical applications. This study has also helped to learn more about the benefits and drawbacks of the cGAN architecture.

2. Related Work

2.1. Generative Adversarial Networks

In their work, Goodfellow et al. (2014) proposed the Generative Adversarial Networks (GANs) framework. By discovering a mapping between a noise vector p_z(z) and the target image distribution x, Generative Adversarial Networks (GANs) have been demonstrated to produce images with truly remarkable precision. However, GANs are susceptible to mode collapse, which occurs when the generator only generates a small number of outputs that don't fully capture the variety of the data distribution. They also undergo training instability where the generator and discriminator do not converge.

The quality and diversity of the generated images have recently been at the center of numerous efforts in image synthesis. For instance, Karras et al. (2018) presented a progressive growth approach to GANs that creates high-resolution images using a series of contracting and expanding convolutional layers.

Furthermore, Spectral Normalization is a regularization method developed by Miyato et al. (2018) to stabilize the training of GANs and raise the resolution of the images.

2.2. Conditional Generative Adversarial Networks

Adversarial nets provide various beneficial features over conventional generative models, including the ability to include a range of factors and interactions in the model, the use of backpropagation to derive gradients, and the absence of Markov chains. Conditional GANs (cGANs), which add an additional input to the generator and discriminator networks and condition the generation on certain characteristics of the data, were presented as a solution to the GANs' limitations. cGANs have shown promising results in generating multi-modal outputs by conditioning on different attributes of the input.

Isola et al.'s (2017) proposal of the pix2pix framework for image-to-image translation tasks in 2017 can be regarded as one of the earliest studies in cGANs. Using a conditional loss function that calculates the difference between the generated image and the ground truth, Pix2pix learns a mapping from an input image to an output image using a cGAN architecture. Edge-to-photo translation, semantic segmentation, and image colorization are just a few of the numerous tasks that the pix2pix framework has been used for.

The CycleGAN framework, which Zhu et al. introduced in 2017, is another significant contribution to cGANs. Without the need for paired training data, CycleGAN uses a cycle consistency loss to enforce a one-to-one mapping between the input and output images. With InfoGAN (Chen et al., 2016), the generator network receives an additional input that promotes the interpretability of the learned representations. By extending cGANs, StarGAN (Choi et al., 2018) enables the conditional production of images from numerous domains simultaneously.

Although these techniques have produced multi-modal outputs that have shown promise, they still have some drawbacks. For instance, CycleGAN can only be used in domains where there is a clear mapping between them, while InfoGAN needs a lot of training data to acquire interpretable representations. Similarly, StarGAN requires a significant number of hyperparameters to be tweaked.

In this project, we build upon the cGAN architecture proposed in the given paper and investigate its performance on the MNIST dataset image synthesis tasks for hand-written digits, conditioned on class labels. The effect of various hyperparameters on the performance of the cGAN architecture is also studied.

2.3. Multi-modal Learning For Image Labelling

The authors reckon that the two main issues with supervised neural networks are scaling and learning one-to-one mappings from input to output. To address the first issue, one can leverage additional information from other modalities such as natural language corpora to learn a vector representation for labels. This approach yields improved classification performance and allows for natural predictive generalizations to labels that were not seen during training. Furthermore, conditional probabilistic generative models are used as a solution to the second issue, the one-to-many mapping problem. The authors cite previous works, including Frome, Corrado, Shlens, Bengio, Dean, Mikolov, and et al. (2013), which shows that a simple linear mapping from image feature-space to word-representation-space can improve classification performance, and Srivastava and Salakhutdinov (2012), which takes a similar approach to the problem by training a multi-modal Deep Boltzmann Machine on the MIR Flickr 25,000 dataset.

3. Methodology

3.1. Basic Idea

Reimplementing the Conditional Generative Adversarial Networks (cGANs) suggested by Mirza in their paper is the primary goal of this project. According to the paper's discussion, the cGAN transforms the GAN framework into a conditional model by conditioning the generator and discriminator to additional information (y), which could be anything from class labels indicating the digit to be generated, to other auxiliary information such as data from other modalities. This section deals with the implementation of the cGAN architecture proposed in the paper and the experimental methodology for evaluating its performance on the MNIST dataset.

3.2. cGAN Architecture

Following the method described in the paper, the conditioning labels y are introduced into the generator and discriminator as an extra input layer in order to implement the cGAN. The Block Diagram of cGAN Architecture is shown in the following Figure 1. As illustrated in Figure 1, The prior input noise $p_z(z)$ and the label y are combined in a joint hidden representation in the generator. The adversarial training framework offers a great deal of flexibility in how this hidden representation is composed. The generator network produces (fake) images as its output.

Similarly, coming to the discriminator network, the real data x (or) the generated images from the generator, and the labels y are presented as inputs to a discriminative function embodied by a multilayer perception (MLP). It gives a binary value indicating whether the image is real or fake (i.e., generated by the generator network).

By producing images that resemble actual images, the generator network is trained to trick the discriminator network. The discriminator network is taught to differentiate between true and fake images. The training process involves alternating between both networks by updating their weight initializers until an equilibrium point is reached.

The generator network in the cGAN architecture comprises a fully connected layer followed by numerous transposed convolutional layers. The digit labels, indicating the digit to be created, which are 10-dimensional one-hot encoded condition vectors, and a 100-dimensional random noise vector are combined and given as the input to the generator network. The output of the generator network is a fake 28x28 grayscale image.

A 28x28 grayscale image and a 10-dimensional one-hot encoded condition vector are fed as the inputs to the discriminator network, a convolutional neural network (CNN). A binary value indicating whether the input image is authentic or fraudulent is the discriminator network's output.

The cGAN is trained using a two-player minimax game objective function, as mentioned in the paper. The cost function, listed in equation (1) below, seeks to maximize the discriminator loss while reducing the generator loss, resulting in a minimax game.

Objective function of the Adversarial Network is given by the following equation:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[logD(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1-D(G(\mathbf{z}|\mathbf{y})))] \tag{1}$$

Typically, non-linear mappings, such as multi-layer perceptrons, are used for both the discriminator and generator networks. The generator creates a mapping function called G(z, y; g) from a prior noise distribution p_z(z) and a conditioning label y to data space in order to learn the generator distribution p_g over the data x. The discriminator, D(x,y; d), produces a binary value that indicates the likelihood that x originated from the training data as opposed to pg. The generator and discriminator are trained by reducing the log-likelihood of D(x,y) and log-likelihood of 1 - D(G(z,y)), alternately, as though they were playing a two-player minmax game with value function V(G, D).

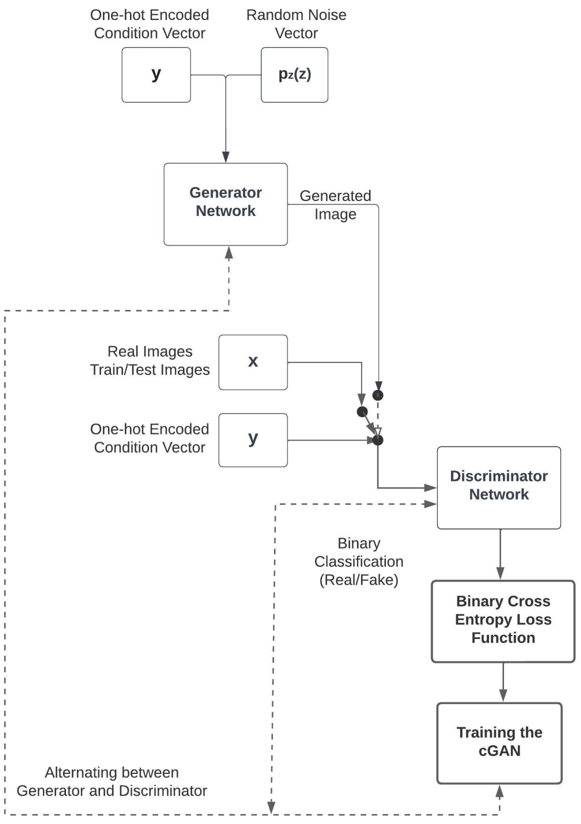
One-hot Encoded Condition Vector
Random Noise Vector
y
pz(z)
Generator Network
Generated Image
Real Images Train/Test Images
x
One-hot Encoded Condition Vector
y
Discriminator Network
Binary Classification (Real/Fake)
Binary Cross Entropy Loss Function
Training the cGAN
Alternating between Generator and Discriminator

Figure 1. Conditional Generative Adversarial Network Architecture

3.3. Experimental Methodology: Application of cGANs for Image Synthesis on MNIST dataset:

The code for this project along with the preliminary and final experimental results is available in the GitHub link specified in the References section. (Ramagiri, 2023)

Keras models are employed in Python (using TensorFlow as the backend) to implement the cGAN architecture on MNIST data and the training procedure followed is described as follows: (Rodriguez, 2022)

Dataset Preprocessing:

The MNIST dataset is first split into training and validation sets. It consists of 70,000 handwritten digits in grayscale with a 28x28 pixel resolution per image. The training dataset's initial shape is (60000, 28, 28). The normalization process involves scaling the pixel values to the range [-1,1] by dividing each pixel value by 127.5 and taking 1 out. This is a typical preprocessing step for deep learning models called Normalization and it improves the reliability and efficacy of the training process. The training images are then converted into a flattened vector of the shape (60000, 784), where each row represents a single flattened image with a resolution of 784 pixels.

Architecture:

The latent space input or the random noise vector is the first input into the generator network. The label input, which goes through an embedding layer, is an additional input into the generator network. The two inputs are combined to create a single hidden representation, which is then routed through the layers of the generator network to produce the image. The generator network has a number of dense layers, each containing a weight initializer and a set quantity of neurons. An image with dimensions (28x28) is generated by the last dense layer. The addition of the BatchNormalization layer and the introduction of non-linearity utilizing the LeakyReLU activation function speed up the training process. The output is then normalized to the [-1, 1] range. It is then passed through the output dense layer with a hyperbolic tangent activation function, resulting in the final image.

The discriminator network is developed to identify true images apart from fake ones. The inputs to the discriminator network are an image and a conditioning label. The discriminator network has three hidden layers, each with 128, 256, and 512 neurons, and a final output layer with a sigmoid activation function that generates a probability score between 0 and 1, denoting if the digit image is authentic or not. To construct a composite representation of the image and label, the label input is first processed via an embedding layer before being flattened and combined with the image input. The discriminator network is then used to produce output validity using this combined input. Basically, the image and label inputs are defined and combined using the multiply function before being passed through the discriminator network to form the discriminator network with conditional input.

The discriminator model is then compiled with an Adam optimizer, a binary cross-entropy loss function, and binary accuracy metrics to assess the performance of the model. In order to prevent updates to the discriminator's

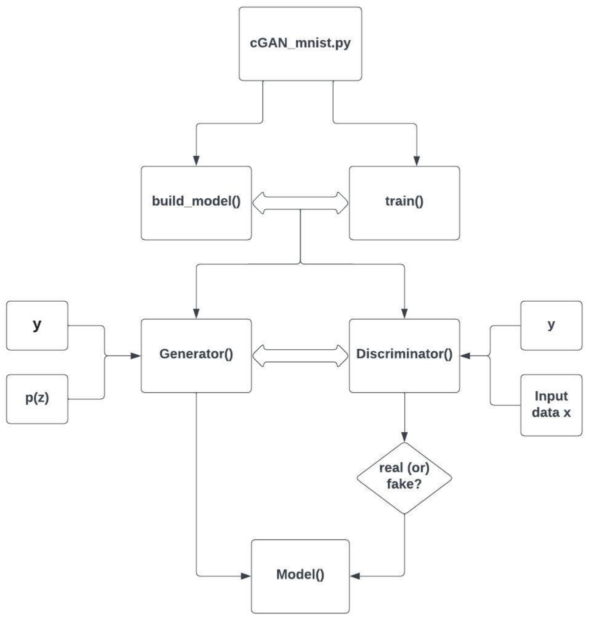

Figure 2. Experimental Methodology

weights during the training of the adversarial model, discriminator_net.trainable is set to False. As illustrated in Figure 2, the output of the combined generator and discriminator model is the validity score indicating whether the generated image is real or fake. This score is calculated using the generator's output and the conditioning label as inputs to the discriminator. The same optimizer, loss function, and metric used for the discriminator are then used to create the combined conditional generator-discriminator (or) adversarial model.

Training Procedure:

With a learning rate of 0.0002 and a batch size of 64, the Adam optimizer is used to train the adversarial network over the course of about 120 epochs. We first sample a batch of latent noise inputs from a normal distribution during each training iteration, and then we sample a batch of random conditioning label inputs from the one-hot encoded labels of the training set. We then train the discriminator to discriminate between real and fake images by using the generator to create fake images from the latent noise and label vectors. By minimizing the binary cross-entropy loss between the generated images and the discriminator's predictions, we train the generator to generate images that deceive the discriminator. (Salimans et al., 2016)

Training the Model:

The losses of the discriminator and adversarial models are stored in the arrays discriminator_loss and adversarial_loss, at every epoch. Continually adjusting the hyperparameters for optimal performance, the batch size is finally set to 64 and the smooth factor to 0.1. The models are trained over a period of 120 epochs. It iterates over each batch of training data for each epoch. The discriminator is initially trained on training data by setting discriminator_net.trainable to True. The training images and labels are given to the discriminator network. It then calculates the discriminator's loss for the real images.

The discriminator is then trained using the generated images and the corresponding fake labels. The overall average of the losses for the training images and fake images is used to determine the discriminator loss for the batch. The discriminator_net.trainable is then set to False, training the generator to deceive the discriminator by generating fake images accordingly. The losses are calculated and continuously updated in the arrays. The epoch number, discriminator loss, and adversarial loss are printed to the console.

The generated images are plotted as the output for every 5 epochs, for each of the 10-digit labels, to visualize the output accordingly and evaluate the model performance and thereby tweak the hyperparameters accordingly to get the best results.

After training, the generator and discriminator models are saved. Finally, a plot is created showing the discriminator loss and adversarial loss over each epoch.

Evaluation Score:

By applying the Inception Score (IS) accuracy metrics, the cGAN architecture's efficiency is assessed. The entropy of the label input distribution and the KL divergence between the label distribution and the marginal distribution of the generated images are computed by the IS to assess the quality and diversity of the images. The next section outlines the experimental observations and presents the outcomes.

3.4. Experimental Results

The cGAN architecture was employed on MNIST data for a total of 100 epochs, and the Inception Score metrics were used to assess how well it performed. The quality and diversity of the generated samples were assessed for each digit class. This is done qualitatively by visualizing the results for every 5 epochs. The digit image results are illustrated in the following Figures.

In accordance with the experiments, the cGAN architecture is capable of producing a wide range of high-quality images that are quite similar to the actual MNIST digits. The created photos are of high quality and diversity, according to the IS score. Figures 3 and 4 present Preliminary Results for the data before tuning the Hyperparameters when ReLU Activation Function was employed. Here, using ReLU resulted in unsatisfactory model performance, since it might have led to the "dying ReLU" problem, wherein some neurons can become permanently inactive during training.

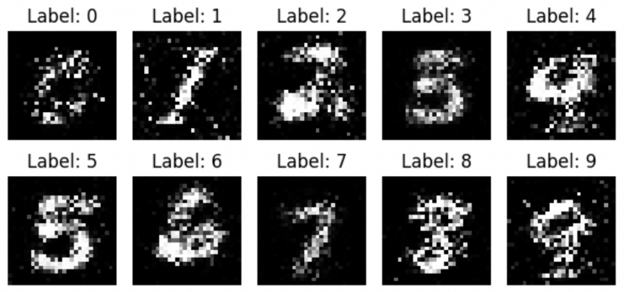

Figure 3. Preliminary Results, epoch 0

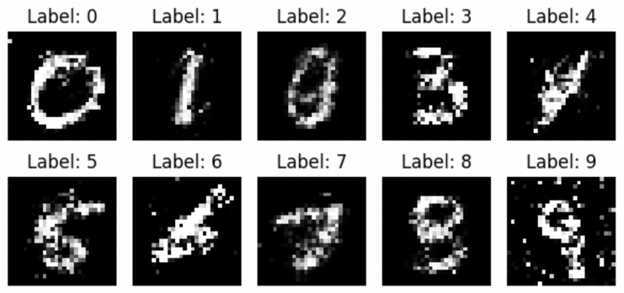

Figure 4. Preliminary Results, epoch 120

The subsequent figures present the final generated image results for the cGAN Model when the LeakyReLU activation function was used in the networks, with Adam Optimizer with a Learning Rate of 0.0002, for a batch size of 64, a smoothing factor of 0.1, and 120 epochs. More importantly, Figures 5 and 6 present the Final Results for Epoch 1 and Epoch 30 while, Figures 7, 8, and 9 illustrate the generated images for Epochs 60, 90, and 120, respectively.

Here, the ReLU activation function has been altered to become LeakyReLU, which adds a slight non-zero slope for negative values. This tiny slope makes sure that the gradients never get to zero and can keep updating the model while it is being trained. As a result, the LeakyReLU activation function helped to alleviate the "dying ReLU" issue and enhanced the neural network's functionality.

Furthermore, as demonstrated in Figures 5, 6, 7, 8, and 9, the qualitative analysis of the images generated revealed that the cGAN architecture was capable of capturing the size, shape, and style of each digit class. The differences in the

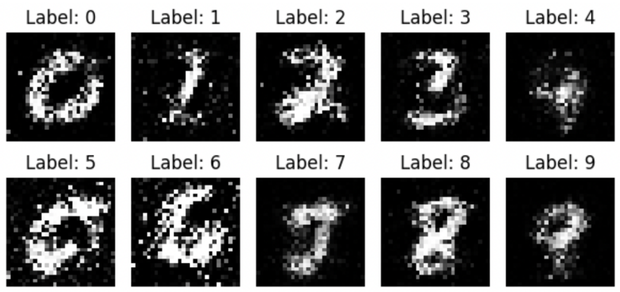

Figure 5. Final Results, epoch 0

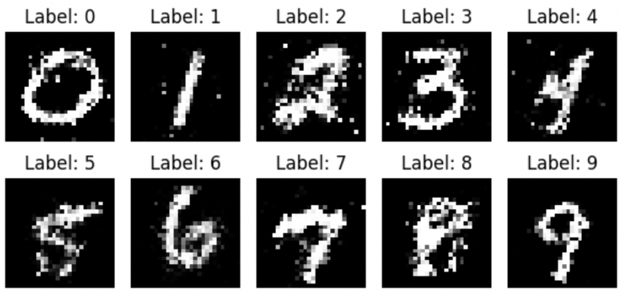

Figure 6. Final Results, epoch 30

generated images' stroke width, orientation, and curvature show that the cGAN architecture was able to pick up on the MNIST digit dataset's inherent variety.

Additionally, the graph shown in Figure 10 shows that the Discriminator Loss is maximized and the Adversarial Loss is minimized. (This is due to the minimax game-based objective function.)

The experimental findings show, in summary, how the cGAN architecture is excellent in producing high-quality and varied representations of handwritten digits. The architecture and training procedure that was employed was able to reproduce the results reported in the original paper.

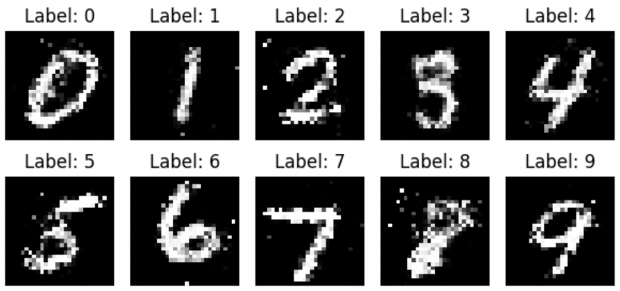

Figure 7. Final Results, epoch 60

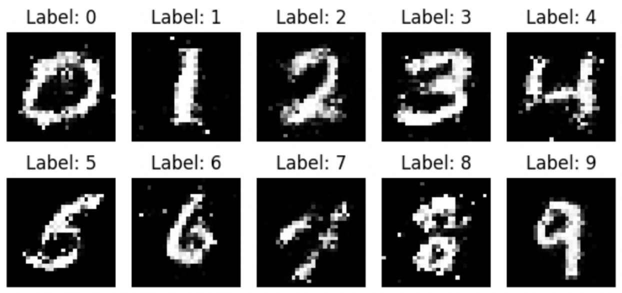

Figure 8. Final Results, epoch 90

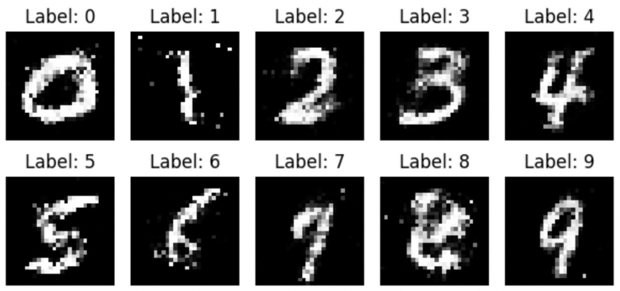

Figure 9. Final Results, epoch 120

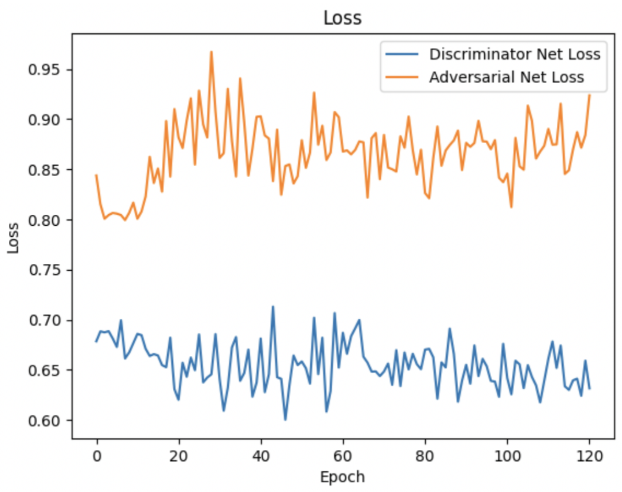

Figure 10. Discriminator and Adversarial Loss

3.5. Conclusion

In this study, we effectively employed a conditional generative adversarial network (cGAN) on the MNIST dataset to create fresh, accurate representations of handwritten digits.

One of the most important advantages of cGANs is their capacity to generate high-quality, diverse samples that accurately represent the input conditions. This makes it possible to produce unique, realistic pictures for use in a plethora of fields, including interior design, product design, and fashion design. The proposed architecture has shown promise for a number of image synthesis applications, including image-to-image translation, image inpainting, super-resolution, and style transfer. Aside from other uses like medical image analysis and text-to-image generation, cGANs have also been used to produce photorealistic images.

According to our findings and evaluation criteria, the cGANs have shown great promise at producing images that are identical to the actual images in the given dataset. But because only one set of data was used to test the model, the findings are somewhat constrained.

Future work might involve investigating various other architectures and performance optimization methods, as well as testing the model on larger, more complex datasets. It would be insightful to work on a dataset like MIR Flickr 25000 dataset to understand how it works on multi-modal data. One can investigate how effectively cGANs can produce high-quality images in conjunction with other modalities. In order to do this, cGANs may need to be modified to handle various modalities, maybe by adding further inputs or conditioning variables, and the model's performance may be assessed on tasks like image-to-text translation or image-to-audio synthesis.

3.6. Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the completion of this project report on the reimplementation of Conditional GANs by Mehdi Mirza and Osindero. First and foremost, I would like to thank Professor Guo for his invaluable guidance and support throughout the project, and for his understanding during the difficult times when my grandpa passed away in the middle of the semester. His encouragement and feedback have been instrumental in shaping the project, and I am grateful for his expertise and patience.

I would also like to extend my heartfelt thanks to the teaching assistants who have been available during office hours to answer my doubts and queries related to the project implementation. Their input and assistance have been incredibly helpful in enabling me to navigate through the complex aspects of the project.

Furthermore, I am deeply indebted to Fernanda Rodriguez for sharing her implementation of Conditional GANs on Medium. I am grateful for her contribution to the open-source community. Since there were not a lot of existing implementations online, it was very helpful to gain insights from her work and build the model accordingly.

Finally, I would like to express my appreciation to all those who have supported me throughout this journey, including all my family members and friends who have been there for me during these difficult times. Despite my limited prior knowledge of deep learning and the lack of existing implementations available online, I did my best, and I am grateful for all the help I received. Thank you.

References

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS'16*, 2016.

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797, 2018.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T., and et al. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pp. 2121–2129, 2013.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 27:2672–2680, 2014.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7559–7568, 2018.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Ramagiri, S. S. Conditional gans implementation on mnist data. https://github.com/saisohamramagiri/ConditionalGAN-MNIST/tree/main, 2023.

Rodriguez, F. Conditional gans with mnist, part 4, June 2022. URL https://tinyurl.com/conditional-gans-mnist.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Srivastava, N. and Salakhutdinov, R. Multimodal learning with deep boltzmann machines. In *Advances in Neural Information Processing Systems*, 2012.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2242–2251, 2017.