DEQ-GAN: Deep Equilibrium Models for Generative Adversarial Networks

Mahathir Monjur
Department of Computer Science
University of North Carolina at Chapel Hill
mahathir@cs.unc.edu

Sam Ehrenstein
Department of Computer Science
University of North Carolina at Chapel Hill
ehrensam@cs.unc.edu

Sharmin Aktar
Department of Computer Science
University of North Carolina at Chapel Hill
sharmin@cs.unc.edu

Zhizhou Liu
Department of Computer Science
University of North Carolina at Chapel Hill
yiwk321@cs.unc.edu

Abstract

GANs (Generative Adversarial Network) are one of the most popular ideas in Machine Learning in the past ten years. The main challenges of GANs are achieving stability, large network size and long training time. To overcome these challenges, we explore the idea of implicit deep learning. In standard deep learning networks, prediction rules are based on a forward, recursive computation through several layers. Implicit rules go much beyond, by relying on the solution of an implicit (or, “fixed-point") equation that is numerically solved, in order to make the prediction. Implicit models have a lot more expressive power than the standard networks. Being inspired by the success of the implicit framework, termed “Deep Equilibrium Models" (DEQ), we have used a DEQ as the Generator Network in GANs to overcome the specified challenges. In this paper, we showed that the DEQ can be used as Generator Network in GANs to generate realistic looking images. We also compared our results with the state-of-the-art GAN networks.

1 Introduction

GAN is about creating, like drawing a portrait or composing a symphony. However, generative tasks are in general more difficult compared to other deep learning tasks [1]. It is much easier to identify a Monet painting than painting one, by computers or by people. But it brings us closer to understanding intelligence. GAN leads us to thousands of GAN research papers written in recent years. In developing games, we hire many production artists to create animation. Some of the tasks are routine. By applying automation with GAN, we may one day focus ourselves on the creative sides rather than repeating routine tasks daily.

In the deep learning research community, interests in Generative modeling have been growing in recent years and it has opened a new hope in the area of unsupervised learning. Currently, the most common generative modeling technique is the Generative Adversarial Network (GAN), where there is a Generator network (G) and a Discriminator network (D). The discriminator (D) is a binary classifier which classifies if the generated data is fake or not and the generator (G) confuses the discriminator by generating realistic data. Thus the Generator and the discriminator play a zero-sum game to improve one another.

There has been a lot of recent work on GANs in various domains including image generation [13], [7], image super-resolution [15], style transfer [25], music generation [11], medical applications [24], text to image generation [19], video prediction [22] and so on. Although GANs have achieved some incredible results by producing realistic images and videos that can fool even human eyes, there are still some challenges. Training of GANs are often unstable, which is even more significant since both the generator and the discriminator are optimized through alternating gradient descent. This instability in training results in problems such as vanishing gradient [2] even with slight changes of the hyper parameters. Additionally, GAN models are often large in size and require a lot of time to train, which is a major hindrance to training a decent GAN that can produce realistic data.

There has been a number of investigations to find alternative of traditional deep learning architectures that would need less memory or training time and has more stability. One of them is deep equilibrium models [4], where the authors presented a new approach to model sequential data based on the observation that most of the existing deep learning models converge towards some fixed point. Computing this fixed point analytically is equivalent to a deep learning model with infinite depth, without the burden of storing the derivatives for each depth during back propagation. Later the authors also applied this new deep equilibrium models in pattern recognition domains [6], specially in image classification and segmentation. They proposed multiscale deep equilibrium models (MDEQ), where they solve for the equilibrium points of multiple feature resolutions simultaneously. The authors showed state of the art performance in the image classification task on the ImageNet dataset and the image segmentation task on the CityScape dataset. They achieved results of comparable quality to a conventional GAN based on ResNet-101, while using a fraction of the memory.

In this paper, we investigate whether deep equilibrium models can be used as a generator of Generative Adversarial Networks and compare the performance with state of the art GANs in terms of performance, training time and memory. However, we do not use DEQ as a discriminator, since discriminator in a GAN setting is nothing but a two-class image classifier and the authors in[6] already showed the effectiveness of DEQ models as image classifier. DEQ models in general are slower than traditional models with explicit layers and hence we only explore the effectiveness of using DEQ as generator in GAN. In summary, our contribution are as follows:

- Redesign and reuse the deep equilibrium model architecture as a generator in GANs
- Measure performance of the new GAN generator in producing realistic looking images on different image dataset such as CelebA [17].
- Compare the performance, training time and memory with state of the art GAN models.

2 Background and Related Work

The basic idea of the deep equilibrium model comes from weight-tying, where all the layers of a deep learning model share the same set of parameters [8]. In general, a weight-tied model has parameter θ on a hidden state z:

$$z^{[i+1]} = f_\theta(z^{[i]}; x), i = 0, 1, ..., L-1$$

These weight-tied deep learning models worked quite well, comparable to traditional deep learning layers [9] [3]. Later, it was found that as we iterate the transformation function f_θ (i.e. add more and more layers), the hidden states tend to converge to an equilibrium (fixed) point, $z^* = f_\theta(z^*; x)$[4]. However, we do not need to find this equilibrium point through iteration, as it would need a lot of time and also we will have to store the derivatives for all the intermediate layers until the equilibrium point is found. Instead, we can formulate fixed-point finding as a root finding problem [4]:

$$g_\theta(z; x) = f_\theta(z; x) - z \implies z^* = \text{Rootfind}(g_\theta; x)$$

This root finding is part of the network architecture itself, and so is carried out whenever the network is used. Any black-box root finding method can be used to find the equilibrium [6].

Unlike conventional networks that need to store the gradients at each layer, we can calculate the loss gradient directly from the inverse Jacobian at equilibrium. Thus, we can update θ with [4]:

$$\theta^+ = \theta - \alpha \cdot \frac{\partial \ell}{\partial \theta} = \theta + \alpha \frac{\partial \ell}{\partial z^*}(J_{g_\theta}^{-1}|_{z^*}) \frac{\partial f_\theta(z^*; x)}{\partial \theta}$$

where $J_{g_\theta}^{-1}|_{z^*}$ is the inverse Jacobian evaluated at the eqiulibrium z^*. As a result, DEQs use significantly less memory, which becomes the bottleneck of conventional deep neural networks as

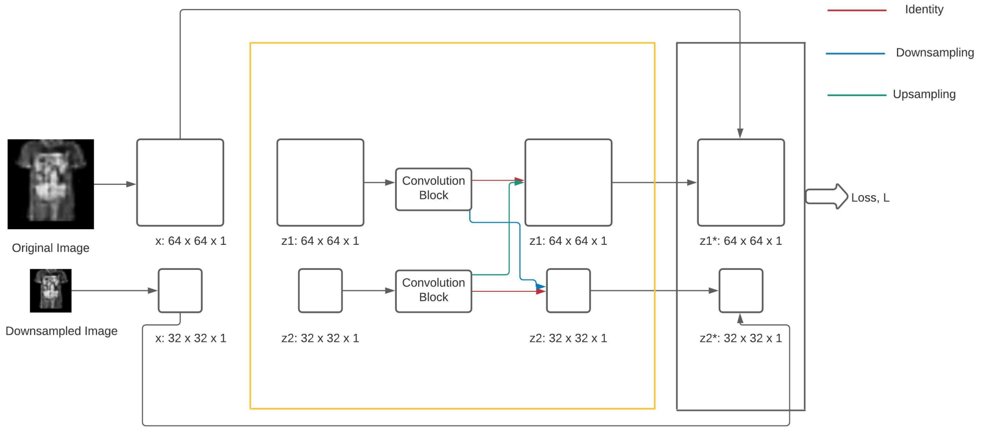

Figure 1: DEQ-GAN model architecture. Note that the elements in the yellow box are the *only* trainable layers; the network's performance comes from the use of the root-finding algorithm on this function. Also, all the convolutional blocks are identical. The sample image used here is taken from the Fashion MNIST dataset [23]

more and more layers are added to them. Using this basic idea of DEQ, the authors in [6] proposed the multiscale deep equilibrium model (MDEQ), where the transformation function f_θ involves several resolutions which are passed through some residual block. At each resolution i, the residual block receives internal state z_i and outputs a new state z_i^+ at the same resolution.

3 Methods

The original DEQ architecture presented in [4] was based on work with trellis networks [5], which are used for language modeling and other sequence-processing tasks. This does not automatically mean that the DEQ will work with images, but on account of the fact that MDEQ was demonstrated to be able to produce good results with semantic segmentation, we chose to start from the MDEQ architecture for our generator. However, we did modify the architecture to fit our purpose, which is to develop a generator of GAN that can produce realistic looking images and videos. The model architecture is shown in figure 1.

In the CelebA dataset, we have images of celebrity faces with resolution of 1024x1024 with 3 color channels, which we downsample to the dimension of 64x64x3. The goal of the DEQ-generator is to generate images of this dimension i.e. 64x64x3. Then, we downsample the input image n times, where n can be of any value, depending on the memory and training time constraint. Our model starts with n hidden states with different resolution- one for each input image. These hidden states are generated randomly, as it is done with any generative adversarial network. These hidden states are then passed through a convolution block, which consists of several Conv2d and ConvTranspose2d layers, so that the resolution of the input and the output of these convolution blocks are matched. These multi-resolution hidden states and the convolution blocks together form our transformation function $f_\theta(z; x)$. Then we compute the equilibrium point of this transformation function, which is the output of our generator. Thus, we get several resolution of the same image as output from our DEQ-GAN model and we combine the losses from all n resolution images when training the discriminator.

Our discriminator architecture is not a DEQ. Instead, we are using a simple convolutional architecture, with two 3x3 convolutional layers each followed by instance normalization and leaky-ReLU with slope -0.2 for values less than 0. We take the highest resolution generator output (64x64x3 channels) as input, and output a scalar critic score.

For training, we chose to use a Wasserstein loss [2] with the Adam optimizer [14]. Wasserstein loss modifies the traditional GAN by turning the discriminator into a critic, the difference being that the critic's output value is an unconstrained score expressing how "believable" the generator outputs are.

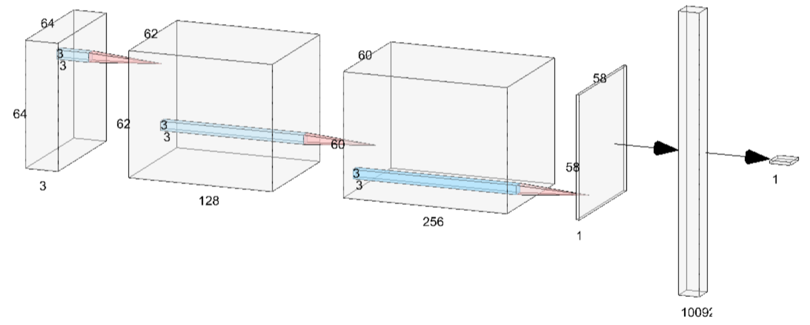

Figure 2: Discriminator architecture. The blue and red arrows are convolution+instance-norm+leaky-ReLU. The first black arrow is flattening to a vector. The second is a linear transformation layer to get a scalar.

This has several advantanges, one being that the discriminator no longer leads to vanishing gradients. With a sigmoid output, a value close to one of the asymptotes corresponds to a very small derivative, which means training will slow down significantly because the discriminator has saturated. Given that discriminating is usually an easier problem, the discriminator in a GAN tends to train much faster than the generator. With Wasserstein loss, this is no longer an issue, as once the discriminator/critic saturates, it essentially acts as a loss function that we can use to continue training the generator. Given that we are using an unconventional, highly experimental generator architecture, Wasserstein loss should remove from the picture some factors that could lead to a lack of results.

Later, we also modified the model to make it a conditional GAN network. We generate an embedding of dimension 64x64x1 from the labels of the images and pass it as an extra channel to the input image. In this way, we can generate images from a certain class, rather than generating images randomly.

3.1 Metrics

Our quantitative performance metric is Frechet Inception Distance (FID). This metric uses the intermediate outputs from the Inceptionv3 network trained on ImageNet as features for computing the distance between two distributions[12]. We computed the FID between the generated images and the training set, which is also the ground truth, and used the PyTorch implementation found at [20]. If $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ characterizes the distribution of the Inception features from the generated data, and $\mathcal{N}(\boldsymbol{\mu_g}, \boldsymbol{\Sigma_g})$ characterizes the distribution of the Inception features of the training data, then the FID is defined as

$$||\mu - \mu_g||_2^2 + \text{tr}(\boldsymbol{\Sigma} + \boldsymbol{\Sigma_g} - 2\boldsymbol{\Sigma}\boldsymbol{\Sigma_g}^{1/2}) \tag{1}$$

In addition, we also measured training time and maximum GPU memory allocated, as these are also important performance metrics, especially for a network architecture like this one which is designed to use less memory at the cost of longer training times.

4 Experimental Setup

In this section, we will describe the setup we used for our experiments. We used the latest version of PyTorch for implementation and the codes can be found here: `https://github.com/Monjur-Mahathir/DEQ-GAN-755-PROJECT`. We used different GPU devices for testing on different datasets, however the same devices were used when comparing the performance of our model with the state-of-the-art WGAN model.

4.1 Dataset

We tested our model on a number of datasets, namely MNIST [10], Fashion MNIST [23], CelebA [16], Anime Face and Pokemon dataset. The MNIST dataset contains 60,000 gray scale labeled images of

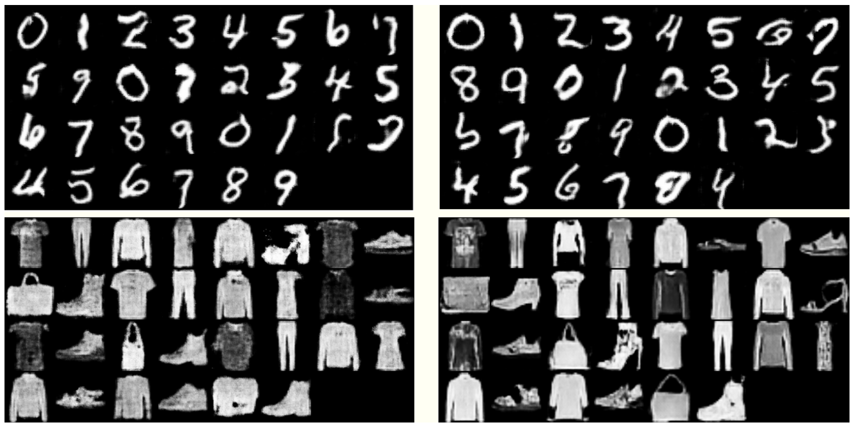

Figure 3: Generated images on MNIST and Fashion MNIST dataset. The top row shows the generated images from MNIST dataset and the bottom row shows the generated images from Fashion MNIST dataset. In both cases, the images on the left are generated from our model and the images on the right are generated from SOTA WGAN model

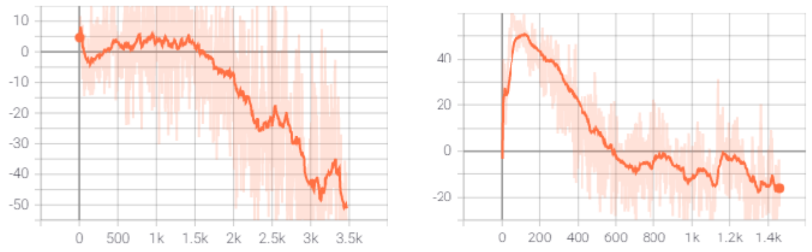

Figure 4: The left image shows the generator loss during training on MNIST dataset and the right image shows the generator loss during training on Fashion MNIST dataset

handwritten digits. Each image has a dimension of 28x28. The Fashion MNIST dataset is similar to the MNIST dataset with 10 classes of different types of clothing. The CelebA dataset is a large-scale face attributes dataset with more than 200K celebrity images, where each image has the dimension of 178×218. [Describe Anime Face and Pokemon dataset and cite them in previous line]. We resized all the images of different datasets to 64x64 during training and generated images of the same dimenstion, and normalized colors to have mean and standard deviation 0.5 for color images.

4.2 Qualitative Results

Some samples of conditional WGAN and MDEQ-GAN on MNIST and Fashion MNIST dataset is shown in figure 3. There are 10 classes in each of the datasets and we show a total three sets of outputs from each class. Visually, it can be seen that our DEQ-GAN model produces images that are very close in quality with respect to the WGAN model. The generator loss of our model is also shown in figure 4 to validate that the generator is converging. However, we tested for 10 epochs on each dataset due to hardware limitation since it takes a large amount of time to train a deep equilibrium model that the traditional model with explicit layers.

We also trained our model on the CelebA dataset and got reasonable results. Some sample generated fake celebrity images can be shown in figure 5. Furthermore, previously we were having some background color issue while generating color images as shown in figure 6. However, we identified and solved the problem. We also trained our model on Anime Face dataset and managed to get descent results. Some sample images can be shown in figure 7.

Finally, we trained our model on Pokemon sprite images and achieved good results, shown below. The dataset consisted of 1324 96x96 pixel sprite images, which were prepared by setting them to

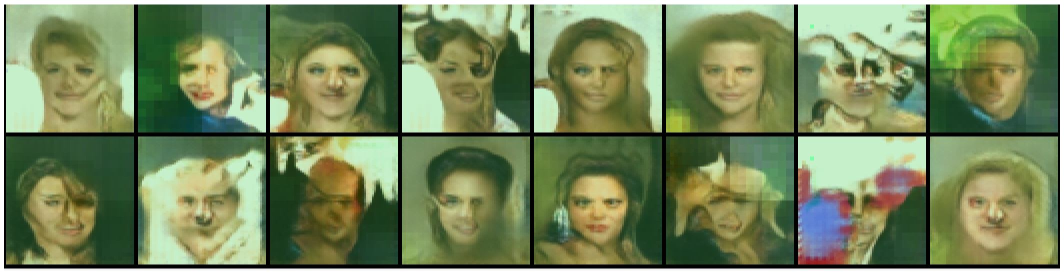
Figure 5: Results on CelebA dataset

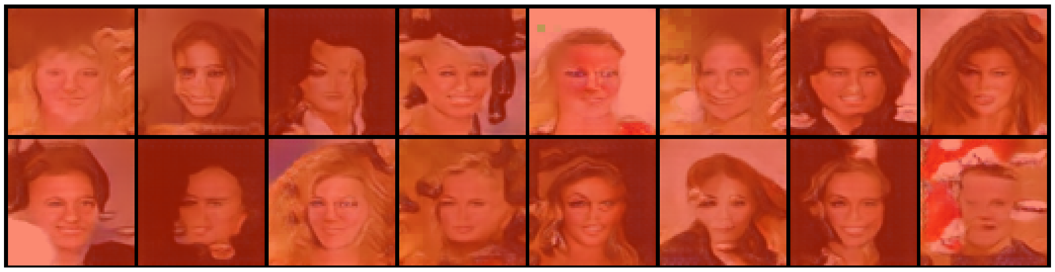
Figure 6: Background color issue on CelebA dataset

have white backgrounds, rescaling them to be 64x64 pixels, and normalizing the colors to have mean 0.5 and standard deviation 0.5. Training was conducted at multiple learning rates and numbers of generator branches (i.e. number of resolutions), in order to study training behavior. Generated Pokemon were evaluated against the training set by using Frechet Inception Distance, pooled to 64 dimensions.

4.3 Quantitative Results

One of the main attractions of using equilibrium models over traditional models with explicit layers is that they require less memory during training. In the original deep equilibrium paper, the authors observed a 2-3x less memory requirement during training than a traditional explicit model with the same number of parameters. We show the resource requirement of our model and compare it with the traditional WGAN model in table 1. As it can be seen that the DEQ-GAN model requires significantly less memory during training than the explicit model such as WGAN. However, the trade-off is the higher execution time. There are some methods to efficiently calculate the Jacobian at the equilibrium point which can significantly improve the execution time. Unfortunately, there were some technical issues which prevented us from implementing that within the scope of this project and leave that as a future plan of this work.

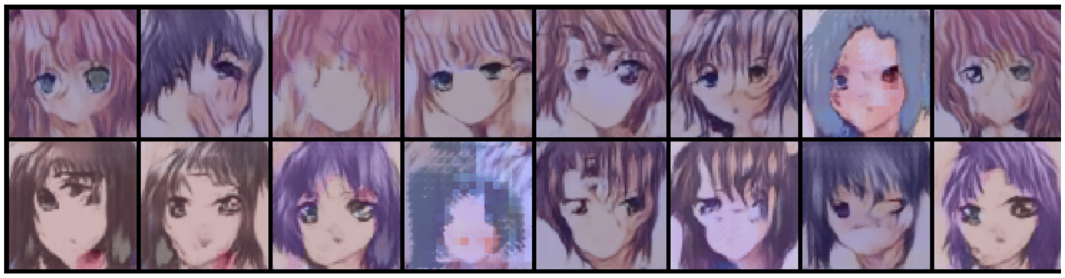
Figure 7: Results on Anime Face

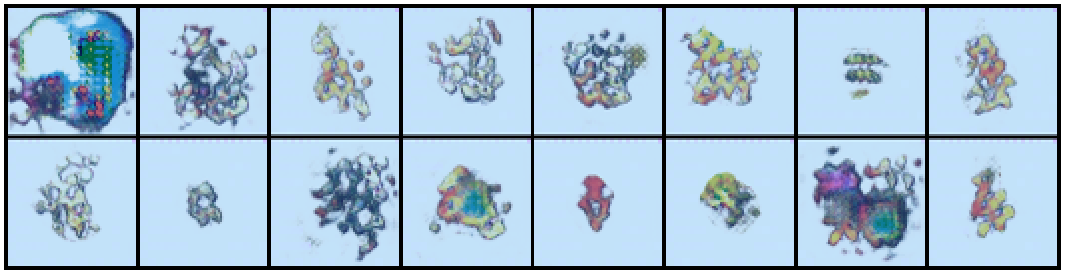

Figure 8: Pokemon results with a learning rate of 1e-4 and 3 branches, trained for 50 epochs.

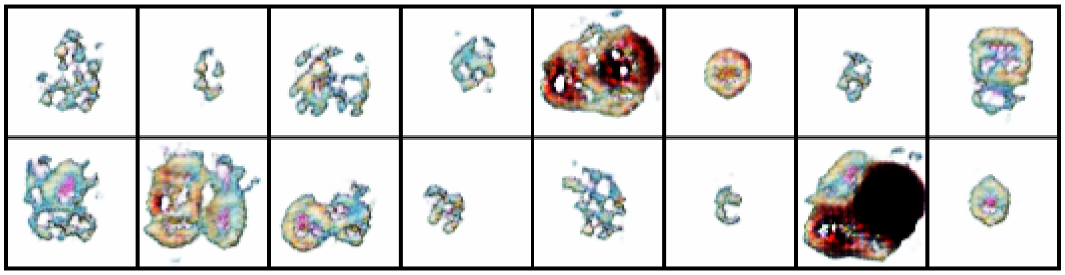

Figure 9: Pokemon generated by training with 1 branch and a learning rate of 1e-4

4.3.1 Pokemon

FID results for Pokemon are shown in Table 2. Our best results were achieved with a learning rate of 1e-4 and 1 branch. The result with WGAN at the same learning rate, number of epochs, and critic update rate had an FID score of 1.142.

5 Discussions and Future Plans

5.1 Analysis of Results

All training datasets showed satisfactory results, with the best being anime faces. This was likely due to the relative complexity: anime faces all have the eyes and hair in the same location. We can see from table 1 that compared to WGAN, DEQ-GAN tended to use approximately half as much memory, but take 4 times as long to train.
On the Pokemon dataset, we see that 1 branch is better than 3. While we would expect including more resolutions in the network would lead to more features being picked up, it seems that this is outweighed by the extra complexity added, leading to the training trying to optimize across multiple resolutions and slowing down. Since we held the number of epochs constant, this led to worse performance.

Dataset	Max Memory Allocated in MB(DEQ-GAN 1 branch)	Max Memory Allocated in MB (WGAN)	Ratio	Execution time of 1 epoch in minutes (DEQ-GAN 1 branch)	Execution time of 1 epoch in minutes (WGAN)	Ratio
Pokemon	1089	2295	0.47	18	4	4.5
MNIST	1281	3117	0.41	48	11	4.36
Fashion MNIST	1139	2937	0.39	52	13	4

Table 1: Comparison of memory usage and training times

	1 branch	3 branches
1e-4	**0.249**	10.546
2e-4	1.539	4.498

Table 2: 64-dimension FID of generated Pokemon for different learning rates and generator sizes, compared to Pokemon sprite dataset, all trained for 50 epochs with batch size 1 and 5 critic (discriminator) updates per generator batch

5.2 Challenges

In our midway report, we stated a goal of generating videos with DEQ-GAN. We were unable to achieve this, due to technical difficulties with training times on still images. These were much longer than expected due to our inability to implement the Jacobian backward-pass training speedup described in [4], which after much investigation we believe is due to a bug in PyTorch or CUDA in our particular use case. In addition, since this is a highly experimental network architecture with few papers or example code bases, we spent a lot of time debugging, probably more than we would have on a better-understood architecture.

5.3 Future Work

Since deep equilibrium models are relatively new, there is a vast opportunity of research in this direction. First of all, the efficient computation of Jacobian at the equilibrium point is required to make the DEQ models faster to train. Secondly, any deep neural network can be theoretically converted to a DEQ model [4] by considering the whole architecture of the DNN as a function of which the equilibrium point needs to be computed. This poses a question of finding out which explicit GAN model works best when converted to an implicit model. In this work, we only considered the WGAN model as the base of our model, but more experimentation is needed to find out the best possible architecture. Even so, our results function as a proof of concept to show that the use of deep equilibrium models in GANs is a feasible direction of research.

We also plan to experiment with the deep equilibrium models to generate videos with fixed number of frames. The preliminary idea is that, we know videos consist of two different kinds of frames- key frames and I-frames [18]. The key-frames are the most informative frames and I-frames ensure the smooth transition between frames of a video. Traditionally, GANs that generate video use an encoder-decoder architecture to encode information from video and then generate frames using the decoder [21]. We plan to design a encoder-DEQ-decoder architecture, where the deep equilibrium network will generate a number of key frames and the decoder will generate the rest of the frames using those key frames. We could not explore this model within the scope of this work, however our plan is to look into this model in the future. Finally, we would also like to experiment with the use of a DEQ decoder in addition to a DEQ encoder. We were unable to do this as we ran into bugs and chose to prioritize other parts of the project.

Overall, we learned a great deal about this new approach to machine learning, as well as about GANs and GAN development beyond what we were able to cover in class.

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*, 2018.

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf`.

[5] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling, 2019.

[6] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *arXiv preprint arXiv:2006.08656*, 2020.

[7] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE international conference on computer vision*, pages 2745–2754, 2017.

[8] Raj Dabre and Atsushi Fujita. Recurrent stacking of layers for compact neural machine translation models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 6292–6299, Jul. 2019. doi: 10.1609/aaai.v33i01.33016292.

[9] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[10] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[11] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models, 2018.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[15] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[16] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

[18] Jiebo Luo, Christophe Papin, and Kathleen Costello. Towards extracting semantically meaningful key frames from personal video clips: from humans to computers. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(2):289–301, 2008.

[19] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis, 2016.

[20] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid`, August 2020. Version 0.2.1.

[21] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[22] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics, 2016.

[23] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[24] Yuan Xue, Tao Xu, Han Zhang, L Rodney Long, and Xiaolei Huang. Segan: Adversarial network with multi-scale l 1 loss for medical image segmentation. *Neuroinformatics*, 16(3): 383–392, 2018.

[25] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 506–511. IEEE, 2017.