COMP 551: Miniproject 4

Lucas Bennett, Bjorn Christensen
McGill University

Reproducibility Summary

Scope of Reproducibility

In this paper, we look to reproduce the findings of Kodali et al.'s (2017) paper "On Convergence and Stability of GANs", which demonstrated the ability for a regret minimization gradient penalty term introduced in their proposed DRAGAN model to provide improved stability as compared to vanilla GAN algorithms. The original paper proposed the idea of studying GAN training dynamics through the form of regret minimization, and how this process would eliminate the prevalence of mode collapse. We looked to reproduce the results of paper that demonstrate this finding, showing how regret minimization effects the stability of loss in generator and discriminator models, and effects inception score.

Methodology

The original code provided by the authors was not sufficient to fully reproduce the experiments and furthermore allowed little expansion on their results. Therefore, we turned to a number of sources to create our own code that would replicate the findings of the original paper. We decided to utilize PyTorch in order to create and evaluate the models. Although the authors used TensorFlow to conduct their experiments, they directed readers to the PyTorch-GAN git repository for a general implementation of DRAGAN using pytorch. The source code of this implementation highly influence the architecture of our recreation. To study inception scoring, we implemented code influence by the PyTorch-Ignite module documentation.

Results

While our results are indicative of the results demonstrated in the original paper, due to computational load, we were not able to demonstrate complete corroboration. This is the case particularly for our findings of inception score. As we were unable to run a model capable of reaching the maximum inception scores highlighted in the paper, it is difficult to assess the added stability of a DRAGAN model solely from our inception score results. On the contrary, our results do demonstrate additional stability in G and D loss in DRAGAN over vanilla GAN algorithms, supporting the results of the paper. We observed the additional stability of DRAGAN in G and D loss on both MNIST and CelebA datasets. As such, although our findings cannot completely validate the original study, they do provide strong suggestions in their support.

What was easy

Creating the discriminator and generator networks were quite simple using PyTorch and training on the MNIST dataset was able to be done using readily available code.

What was difficult

Interpreting the authors methodology in order to follow their pipeline using different packages was quite difficult. The DRAGAN and inception scoring implementations were not compatible, meaning we had to write a large portion of the code from scratch in order to integrate the steps. Furthermore, the code provided by the authors was only suited to analysing the MNIST data-set. As we were far more interested in datasets with RGB images, the need to translate the code to suit multiple channels and convolutional neural networks was also a challenge.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

1 Introduction

Generative adversarial networks (GANs) are foundational models in image generation. Their standard structure consists of two competing networks, a generator and a discriminator, engaging in a zero-sum game. However, the non-convex nature of this classical architecture has produced numerous theoretical challenges which have proved difficult to overcome. One such challenge is the prevalence of *mode collapses*, a phenomena in which the network converges to a local-equilibria (modes), and the generator gets trapped without the discriminator being able to remove it. In their 2018 paper "On Convergence and Stability of GANs", Kodali et al. (2017) proposed a model possessing a novel training method - which they called DRAGAN (Deep Regret Analytic Generative Adversarial Networks) - that could overcome the challenge of mode collapse. It entails the addition of a gradient penalty during the learning process which aims to pull the model out of local equilibria. The validity of their findings are investigated here.

2 Scope of reproducibility

In this paper, we look to reproduce the findings of Kodali et al.'s (2017) paper "On Convergence and Stability of GANs" The two major claims from that paper that we are looking to investigate are,

- Claim 1: Introducing a gradient penalty into the classic GAN training procedure will reduce model instability.
- Calim 2: The inception score of a DRAGAN model will be more robust than the inception score of a vanilla GAN model.

3 Methodology

For MNIST, we used a PyTorch DRAGAN implementation which was constructed based on the author's original tensorflow implementation. We made some slight modifications to the code for ease of use but generally it is identical. We did not conduct an inception score experiment on the MNIST dataset due to computational load, instead we modified the external code in order to analyze the dynamics of the D and G loss. For experiments on the CelebA dataset, we constructed our own code and DRAGAN implementations. This was done as the author had not provided code for inception scoring convolutional models. Our code was highly influenced by implementations and examples in the PyTorch documentation.

3.1 Model descriptions

Both the DRAGAN and vanilla GAN models used identical discriminator(D) and generator(G) networks. For tests with the MNIST dataset, both the D and G networks consisted of a single hidden linear layers. For tests with the CIFAR10 and CelebA datasets, the D and G networks consisted of 4 hidden convolutional layers with ReLu activation, resulting in 10,949,504 parameters in the G network and 2,765,568 parameters in the D network (all trainable). The models were not pretrained; train was done using an ADAM optimizer. Training of the DRAGAN model different due to the inclusion of a regularization term applying a gradient penalty.

3.2 Datasets

The Datasets used were MNIST and CelebA. The MNIST dataset contains black and white images of hand-drawn numbers ranging from 0-9. The images are composed of 28X28 pixels. The distribution of each category in the MNIST training set ranges from 5421 to 6742 occurences, totaling 60,000 total images, which means we have a good spread to learn each number from. Furthermore the numbers in the train set were derived from a separate group of writers from the test set, about 250 for each. The CelebA set includes 202,599 photos selected from 10,177 celebrities. These images are described by 3 64x64 matrices (pixel dimensions), with each matrix representing an RGB channel.

For the MNIST dataset we used a 60,000-10,000 train-test split. CelebA uses a 162,770-19,962 train-test split.

No preproccessing was done on the MNIST dataset. In order to preprocess the CelebA dataset, we normalized the pixel values of our images from 0-255 to 0-1 and centered the images using torchvisions transform utility.

MNIST: `https://www.tensorflow.org/datasets/catalog/mnist`

CelebA: `https://www.kaggle.com/datasets/jessicali9530/celeba-dataset`

3.3 Hyperparameters

We chose the hyperparameters using manual search to find those that would provide sufficient qualitative results without requiring extensive run times. For training of the CelebA dataset, we found smaller small learning rates (0.0001) were required to generate sufficient results. Larger learning rates blew up d-loss and resulted in the extinction of g-loss. Conversely, smaller learning rates were deemed too slow. We also manual chose the size of the latent space to be 1000. This was done by finding a balance between the decreasing speed of training and increasing model performance as we increased the size of the latent space. We inherited the lambda gradient penalty parameter set in the original paper (lambda = 10). We found a batch size of 32, produced good results while maintaining fast experimental speed.

3.4 Experimental setup and code

Experiments on the MNIST dataset consisted of a standard mini-batch stochastic gradient descent algorithm to train the vanilla GAN and DRAGAN models, which was modified to record the G and D loss of every iteration during training and visualized the fake image that had been generated. We then plotted the results of the G and D losses using Matplotlib.

The setup of our experiments on the CelebA dataset were more sophisticated. We define two training step functions for vanilla GAN and DRAGAN models. These consisted of a gradient descent optimization step on both the G and D networks, with and without gradient penalty respectively. We then sequentially initialized a PyTorch-Ignite engine with the each training step, initializing new G and D networks and optimizers. We defined a series of operations that would be conducted as the engine trains a dataset, generation of fake images on a set of 64 fixed noise samples very 500 iterations, and recording D and G loss each iteration. This allowed us to analyze the dynamics and the performance of the model quantitatively and qualitatively throughout training. After every epoch of train, we recorded the FID and inception score of the model.

3.5 Computational requirements

Our experiment on the MNIST dataset was done across 1000 iterations of training conducted on the apple M2 CPU, which required roughly 2 hours per model. We do not recommend using a CPU to run these experiments, and would suggests using a device with CUDA support.

Experiments on the CelebA dataset were done using a cuda0 GPU with a nccl parallel backend. Each epoch of training on all batches required under 4 minutes (training batch size of 64), although it varies with batch size. Evaluation of FID and inception score after each epoch took roughly 2.5 minutes when using a test-batch size of 256. Increasing the size of the test batch greatly increased the speed of our experiments without altering results. We do not suggest greatly altering the batch size of the training set as it minimally reduces experimental time, but has a large influence on results (see additional results 2).

4 Results

Our results found faster rates of convergence in G loss for the DRAGAN model in both the MNIST and CelebA datasets as compared to the vanilla GAN model. Furthermore, there was far less spread in both G and D loss when using DRAGAN, supporting the claim of the original paper that DRAGAN would increse the stability of the model. However, we found the inception score of the DRAGAN to be generally reduced as compared to the inception score of vanilla GAN. It is of note that the inception score of DRAGAN did show consistent increase whereas the inception score of the vanilla GAN showed a sharp decrease followed by a recovery. These results do suggest the results of the original paper, however, we would need more iterations to sufficiently reproduce them.

4.1 Results reproducing original paper

4.1.1 Result 1

In this experiment, we found analyzed the dynamics of G and D loss of the vanilla GAN and DRAGAN models on the MNIST and CelebA datasets. The results supports the first claim of the paper, suggesting that DRAGAN improves model stability. We were able to successfully reproduce these results on both the MNIST dataset and CelebA dataset as shown in figure 1 and 2 respectively.

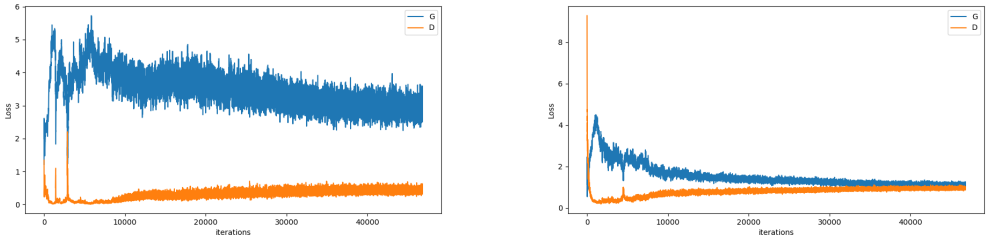

Figure 1: (MNIST) G and D loss for vanilla GAN (left) and DRAGAN (right) models.

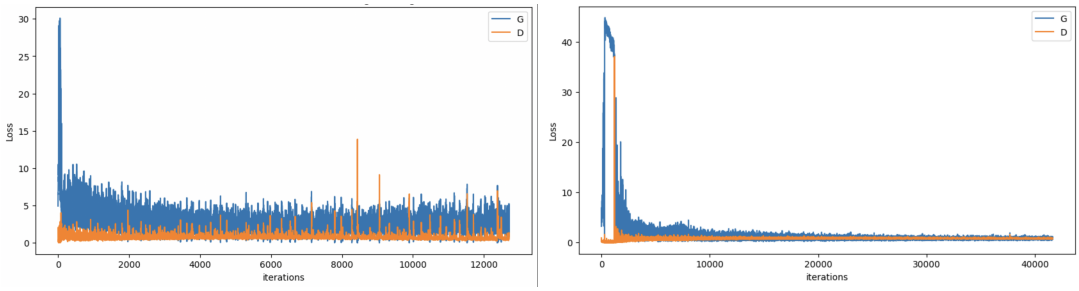

Figure 2: (CelebA) G and D loss for vanilla DSGAN (left) and DRAGAN (right) models.

We see faster convergence and limited spred of G loss when we introduce the regualarization term to the model.

4.1.2 Result 2

These results demonstrate the inception score of vanilla GAN and DRAGAN at various epochs. The number of epochs we were able to study is limited by computational need, however, the robustness of the inception score for the DRAGAN model suggests support of claim 2.

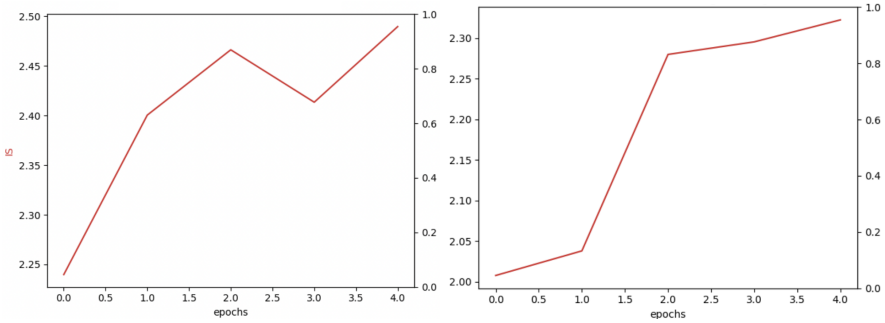

Figure 3: (CelebA) Inception scores for vanilla DSGAN (left) and DRAGAN (right) models across various epochs of training.

4.2 Results beyond original paper

4.2.1 Additional Result 1

We wanted to provide additional qualitative results to those presented in the paper. To do so, we generated a set of 64 fixed noise latent space samples and visualized the generated images of the models after every 500 iterations. A GIF of the models progress is saved with our code.

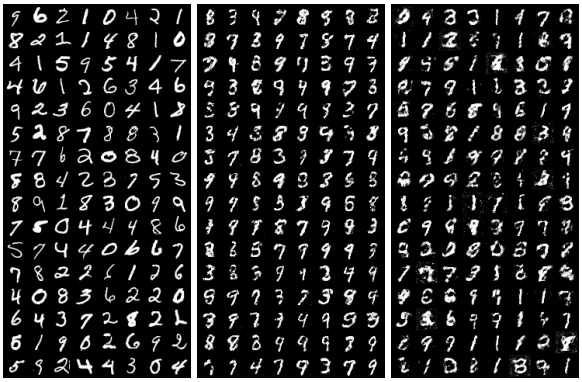

Figure 4: (MNIST) Real(left) and generated samples from vanilla GAN (middle) and DRAGAN (right)

Figure 5: (CelebA) Comparison of real and generated images for DRAGAN (top) and vanilla GAN (bottom)

In figure 4 and figure 5, we see that the vanilla GAN model is producing more accurate depictions of the numbers and faces. This is an interesting finding which may not be expecting with the results of the original paper. It is of note, that

the vanilla GAN appears to be overproducing images of one type. For example, there is a strong over representation of the number nine in the MNIST generated images. This could be indicative of mode collapse.

5 Discussion

While our results are indicative of the original papers claims, due to computational load, we were not able to demonstrate complete corroboration. This is the case particularly for our findings of inception score. As we were unable to run a model capable of reaching the maximum inception scores highlighted in the paper. Furthermore, we were limited in the number of epochs we were able to record the inception score, therefore, we cannot say with certainty that the DRAGAN algorithm prevents mode collapse. However, the variance observed in the inception score of vanilla GAN opposed to the steady increasing inception score of DRAGAN is promisng. On the contrary, our results do demonstrate additional stability in G and D loss in DRAGAN over vanilla GAN algorithms, supporting the first claim of the paper. We observed the additional stability of DRAGAN in G and D loss on both MNIST and CelebA datasets. As such, although our findings cannot completely validate the original study, they do provide strong suggestions in their support.

5.1 What was easy

Creating the discriminator and generator networks were quite simple using PyTorch and training on the MNIST dataset was able to be done using readily available code.

5.2 What was difficult

Interpreting the authors methodology in order to follow their pipeline using different packages was quite difficult. The DRAGAN and inception scoring implementations were not compatible, meaning we had to write a large portion of the code from scratch in order to integrate the steps. Furthermore, the code provided by the authors was only suited to analysing the MNIST data-set. As we were far more interested in datasets with RGB images, the need to translate the code to suit multiple channels and convolutional neural networks was also a challenge.

References

Kodali,N. Abernethy, J. Hays, J. Kira, Z. (2017). On Convergence and Stability of GANs.