Anas Hamed - ahamed@buffalo.edu - Person #: 50295901
Instructors: Sargur Srihari - Mohammad Shaikh - Marissa Dominijanni
23 October 2019

Conditional Self Attention Wasserstein Generative Adversarial Networks

Introduction

Generative Adversarial Networks (GANs) have gained significant popularity in the deep learning field over the recent years. GANs have proven to be useful in a wide range of applications, which was not the case just a few years ago when the idea for GANs was first conceived. It was the case that these generative networks were very difficult to train, did not sufficiently capture the true distribution of the data on which they were trained, and often did not converge. Owing to recent developments in the methods and architectural components used in training these networks, results have improved drastically.

Among the methods used to improve the performance of GANs, the following will be discussed:

- Self Attention
- Wasserstein Loss
- Spectral Normalization
- Conditional Training

These features were implemented and trained on the CIFAR10[1] dataset. The results of the implementation will be provided and contrasted to an "unembellished" Deep Convolutional GAN (DCGAN[2]).

Background

This section aims to shed light on the concepts that were mentioned above so as to realize what their contribution to the enhancement of the GAN's performance comes down to.

1. Self Attention: conceived by the Google Brain research team[3], the idea of self attention is to force the network to "pay attention" to specific parts of each datum, and give more importance to locations within the sequence of points in the datum that better represent the features that the network is trying to learn. This is a departure from the previous approach of relying on convolutions which typically only take into account local regions

[1] https://www.cs.toronto.edu/~kriz/cifar.html
[2] https://arxiv.org/abs/1511.06434
[3] https://arxiv.org/abs/1706.03762

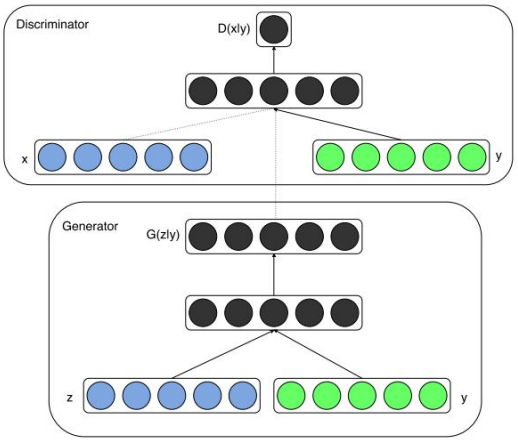
within each sample. The problem with the previous approach can be realized in cases where certain dependencies exist in the sample that are separated by a large distance. Self attention alleviates this issue by constructing a feature map at a certain layer in the architecture, then feeding that feature map to the succeeding layer. Self Attention has the effect of shifting the weights in favor of those more representative features.

2. Wasserstein Loss: compared to DCGANs which typically use Binary Cross Entropy loss, the Wasserstein loss can be of great help in the issue of mode collapse. The objective of the function is to minimize the Earth Mover (EM) distance between the real data distribution, and the fake data distribution. The loss is characterized by the following formula:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right]$$

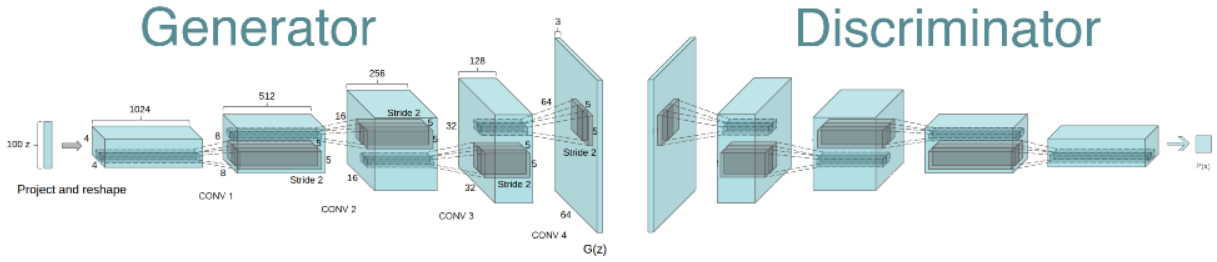
3. Spectral Normalization: this is a normalization technique of data that renders both the discriminator and generator functions of the GAN Lipschitz continuous. By imposing this constraint, the functions of the GAN will have bounded gradients, which eliminates the problems of vanishing and exploding gradients. Simply put, the process is carried out by dividing the matrix that represents the weights of the layer on which the normalization is applied by the spectral norm of the matrix. The spectral norm of a matrix is simply the largest singular value of that matrix.

4. Conditional Training: this is a simple yet effective technique wherein the class of the sample on which the networks train is fed along with the sample itself. For the generator, the class is appended to the noise vector which is used to generate a sample, and for the discriminator, it is injected at a fully connected layer that follows the convolutional layers that constitute its layers. The figure below depicts the process. The *y* vector represents the label, *x* represents the input, and *z* represents the noise vector. Conditioning has the effect of bringing out the features that represent the class that was passed in for both the generator (*G(z|y)*) and discriminator (*D(x|y)*).



Implementation

The aforementioned features were implemented by taking the DCGAN architecture and training cycle, and making changes to it. The diagram below illustrates the original architecture of the DCGAN, which will be referred to in the steps that follow.



The changes that were made are as follows:

- To implement Self Attention, an attention layer was placed after the 2nd convolutional layer in the discriminator, and after the 3rd transpose convolution layer in the generator. As was shown in the paper[4] that introduced Self Attention as a method to overcome the shortcomings of the convolution, the steps involved in applying the process are as follows:
    - Take the output of the preceding convolutional layer as input to the SA layer
    - Convolve the input with a 1x1 kernel three separate times to produce query *q,* key *k,* and value *v*
    - Perform a matrix multiplication between *q* and *k*, then apply a softmax operation on the output. The output of this operation yields the attention map
    - Perform yet another matrix multiplication with the attention map and the value v, and that produces the final output of the SA layer

- To implement the Wasserstein loss, the activation function of the discriminator was replaced with a linear activation layer. The cost function for the generator was changed to be the mean of the prediction values of the discriminator instead of a Binary Cross Entropy (BCE) loss. The cost function of the discriminator, which was also a BCE loss in the DCGAN, was also replaced with the Wasserstein loss below. Note that the penalty terms at the end of the equation represents a penalty on the gradient that constrains the function as described in the previous section.

$$\mathfrak{L}(D) = (Prediction_{fake} - Prediction_{real}) + ??_{penalty}$$

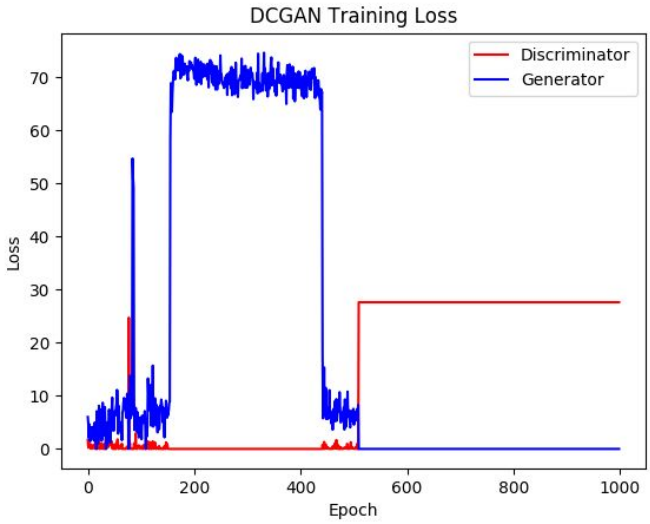
[4] https://arxiv.org/abs/1706.03762

- To implement Spectral Normalization, each convolutional and transpose convolutional layer in both the discriminator and generator had the normalization function applied to their weights prior to being processed by the layer.
- To implement conditioning in the GAN network, a one-hot encoded vector that represents the class of each sample was fed in along with the 100-dimensional noise vector for the generator. For the discriminator, the output of the last convolutional layer was concatenated with the same one-hot encoded vector of the class that was first fed to a fully connected embedding layer.

Training

- **DCGAN**

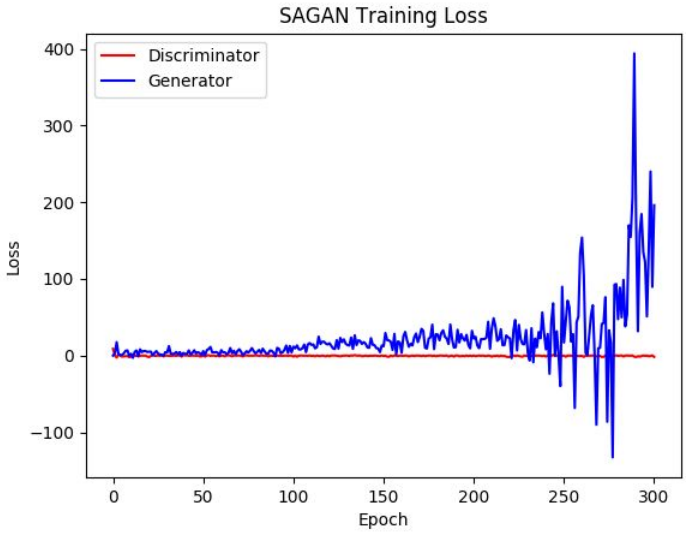
The DCGAN was trained for 1000 epochs with a batch size of 64. The training loss and FID scores for the DCGAN are shown below:



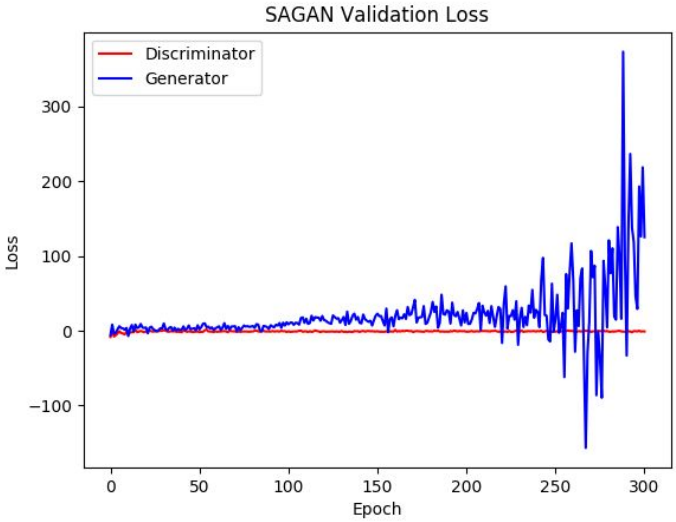
As the training graph shows, for the first half of the training process, the losses for the two networks remained about the same until there was a sudden increase in the loss of the generator, and a gradual decrease in that of the discriminator. In the second half, it can be seen that the discriminator's loss plateaus around 27, and so does the generator's around 0. It is obvious that the GAN's networks converged in the second half.

- **CSAWGAN**

  The graph below shows the training curve for the CSAWGAN, trained for 300 epochs with a batch size of 64. As the figure shows, the discriminator's loss hovers around zero, while that of the generator oscillates wildly towards the end of the 300 epochs, but remains within a range of about 20 from the discriminator's loss. The graph suggests that the model has not yet converged, which is probably because 300 epochs are not sufficient to reach convergence.



The figure below shows the loss values for a validation set of the same batch as that of the training set. The graph shows that the curve follows the same trend as in the training over the 300 epoch period. This seems to suggest that the network is not overfitting, and is thus learning the true distribution of the data.

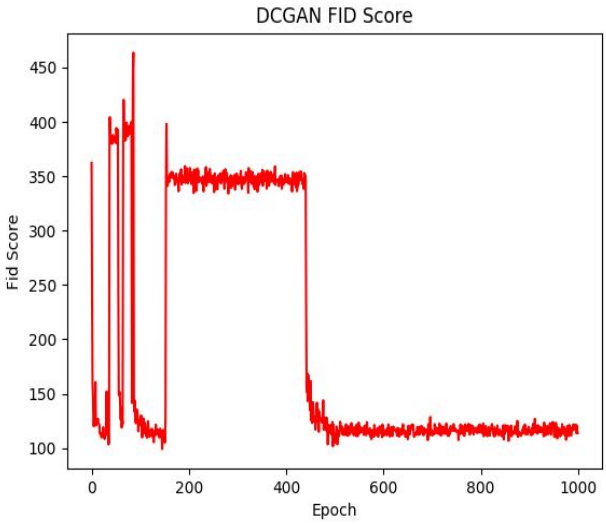


Results

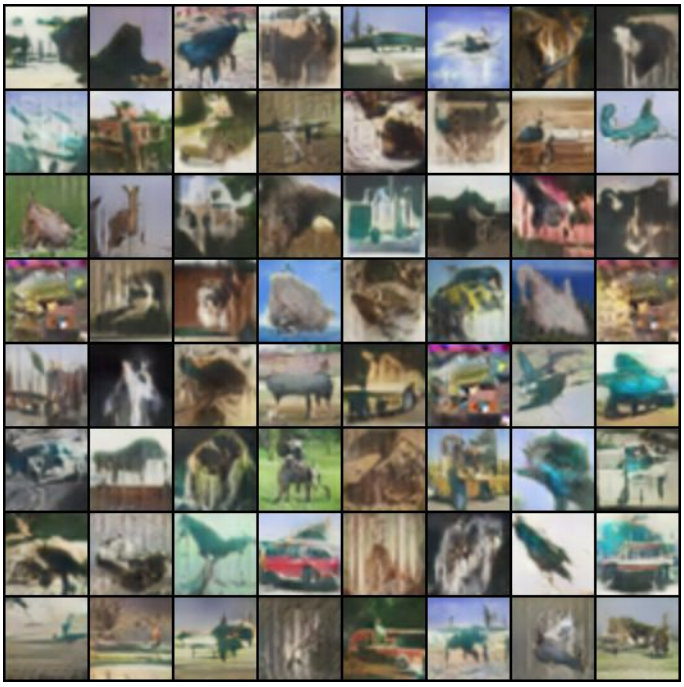
The results will now be discussed in terms of how realistic the images look, as well as the Frechet Inception Distance for the two GAN at different epochs.

- **DCGAN**

For the FID score, there were fluctuations in the first half, but a best score of around 100 was achieved throughout the epochs. Scores as low as 5 were achieved in other state-of-the-art architectures, so this score is not particularly good.



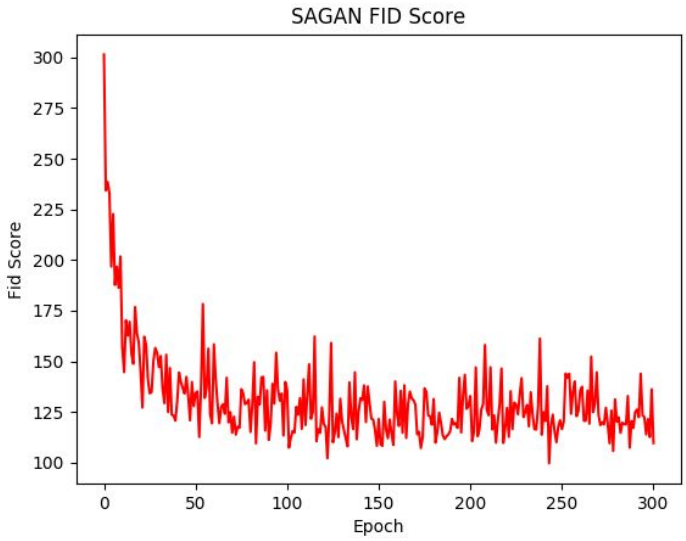
The figure below shows one of the samples that were generated from one of the lowest scoring FID epochs of the generator:



The network did produce some plausible looking images, but it is easy to recognize these images as fakes, since there is a lot of distortion and nonsensical features in some images.

- **CSAWGAN**

The graph below shows the FID scores calculated for the 300 training epochs. As the graph shows, the fid scores decreases steadily and in a more gradual fashion than the DCGAN's score did. This is may be spectral normalization at work, since it is intended to make the gradients more well-behaved and eliminate exploding/vanishing gradients. While the score itself is still high compared to that of DCGAN, it must be noted that CSAWGAN only trained for about a third of the epochs that the DCGAN trained on.

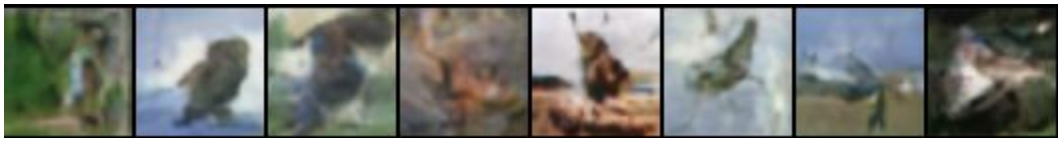


Since conditioning was implemented in the CSAWGAN, we can generate images of certain classes by giving the generator the class we want to generate. The following images were taken by providing a different label each time:

- Class 0: airplane

- Class 1: automobile

- Class 2: bird



- Class 3: cat

- Class 4: deer

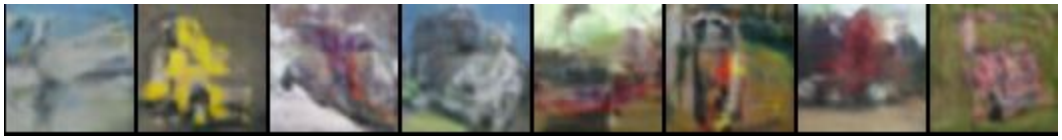
- Class 5: dog

- Class 6: frog

- Class 7: horse

- Class 8: ship

- Class 9: truck



Conclusion

As the results of the CSAWGAN may not be significantly better than the ones obtained by DCGAN, this may be due to insufficient training or an ill suited hyperparameter. But even these results show the potential and influence of the added features. Some classes are much better represented in the CSAWGAN than in the DCGAN, which, by itself, makes it much more powerful than the latter.