Generative Adversarial Networks (GANs)

Project 1

CSE676: Deep Learning(Fall 2019)

Esther Raja Kumari Katti
50288205

1 Introduction

The aim of this project is to learn a generative model from which samples can be learned.For this project I am implementing two types of GANs: Deep Convolution GAN and Conditional Self Attention GAN.We are training the model on CIFAR10 dataset which consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.The Deep Convolution GAN(DCGAN) consists of two deep convolutional neural networks: a discriminator (D) and a generator (G) which compete with each other making each other stronger at the same time.The limitations of DCGAN are overcomed by adding a self attention module in the DCGAN.

2 Model and Training-DCGAN

2.1 DCGAN

The idea of GAN is given a training data,generate new samples from the same distribution.GAN addresses density estimation a core problem of Unsupervised learning.GAN consists of two neural networks Generator network which tries to fool the discriminator by generating real-looking images and dicriminator network which tries to distinguish between real and fake images.In the DCGAN both generator and discriminator are based on Deep Convolutional Neural Network.

2.2 Training DCGAN

The guidelines mentioned in the paper were used for designing and training of DCGAN.
•Replace any pooling layers with strided convolutions.
•Use batchnorm in both the generator and the discriminator. ⌢
• Use ReLU activation in generator for all layers except for the output, which uses Tanh.
• Use LeakyReLU activation in the discriminator for all layers.
No pre-processing was applied to training images besides scaling to the range of the tanh activation function [-1, 1]. All models were trained with mini-batch stochastic gradient descent (SGD) with a mini-batch size of 32. All weights were initialized from a zero-centered Normal distribution with standard deviation 0.02. In the LeakyReLU, the slope of the leak was set to 0.2 in all models. The model performed well when the learning rate was set to 0.0002 and momentum term β to 0.5.

3 Model and Training Self Attention GAN

3.1 Self Attention

The size of the DCGAN model being small prevents it from learning Long-term dependencies.If we try to increase the filter size or the depth of the network the computational costs increases.To overcome this issue a self attention module is assigned in the DCGAN.Self attention helps in finding the correlation between every pixel with all the other pixels even for the regions far apart it can capture the dependencies.

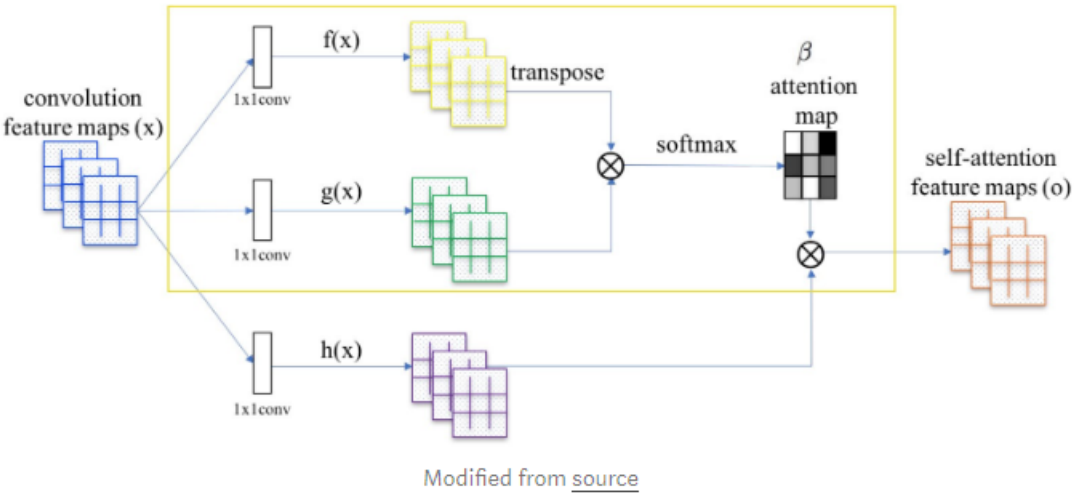

Figure 1: Self Attention

Mathematically the convolutional feature map x is branched out to three different copies.

$$f(X) = W_f x$$

$$g(X) = W_g x$$

$$h(X) = W_h x$$

Then we apply the dot-product attention to output the self attention feature maps.

$$y = softmax(f^T g)$$

$$o = \sum \alpha h$$

Furthermore, the output of the attention layer is multiplied by a scale parameter and added back to the original input feature map:

$$y = x + \gamma o$$

3.2 Conditonal

In an unconditioned generative model, there is no control on modes of the data being generated.In the Conditional GAN (CGAN), the generator learns to generate a fake sample with a specific condition rather than a generic sample from unknown noise distribution.In the generator we concatenate the noise data with labels.In the discriminator we do the same but at the first fully connected layer after the convolutional layer.

3.3 SpectralNormalization

Spectral Normalization is added to stabilize the training of discriminator.Applying spectral normalization to the discriminator network constrains the Lipschitz constant of the discriminator by restricting the spectral norm of each layer

3.4 Experience Replay

The WGAN solves the issue of mode collapse and generator can still learn even when discriminator is performing well.But the The difficulty in WGAN is to enforce the Lipschitz constraint.For that we

can perform Weight clipping. Clipping is simple but the model may still produce poor quality images and does not converge, in particular when the hyperparameter c is not tuned correctly.
Other approach we can use is Gradient Penalty .It penalizes the model if the gradient norm moves away from its target norm value 1. However, gradient penalty adds computational complexity that may not be desirable but it does produce some higher-quality images. I have used third approach called Experience Replay.In this method every now and then, we show previously generated samples to the discriminator and hence prevent the generator from easily fooling the discriminator.

3.5 Training-SAGAN

Implementation of Wassertein loss did not give better results than Hinge loss.I have tried to train the model with the parameters defined in the paper but after few iterations the images were not improving.Adding the attention at 8*8 produced better results compared to adding after 16*16.Though the model has not converged there was a decline in the FID score. In the LeakyReLU, the slope of the leak was set to 0.1 in all models.The model performed well when the learning rate was set to 0.0004 and momentum term β to 0.5.

4 Training Curves

4.1 DCGAN

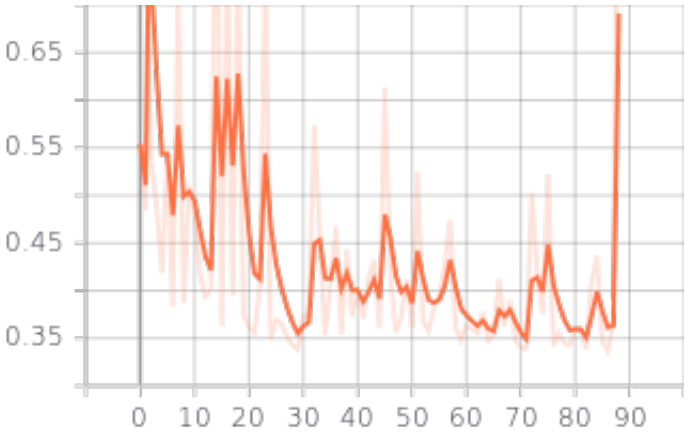

Figure 2: Discriminator Loss for real data

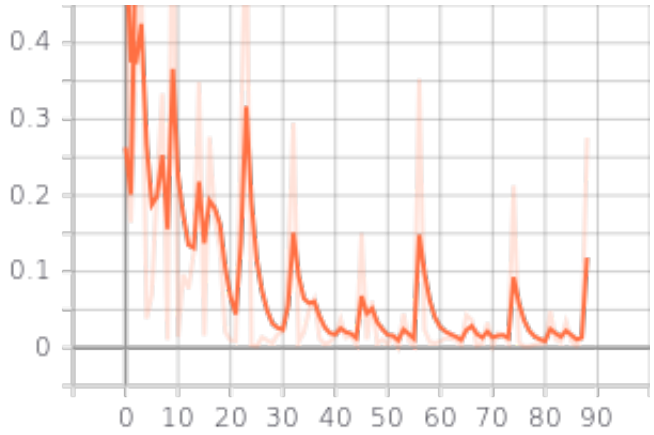

Figure 3: Discriminator Loss for fake data

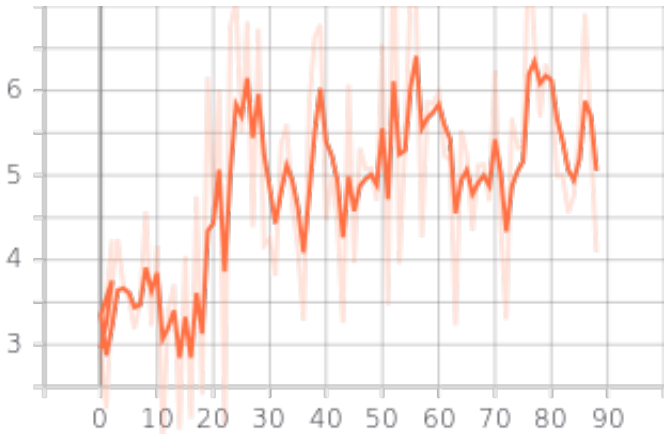

Figure 4: GAN loss

4.2 CSAGAN

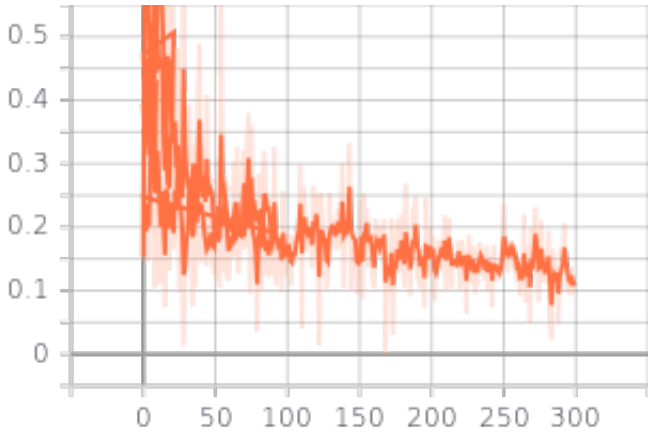

Figure 5: Discriminator Loss for real data

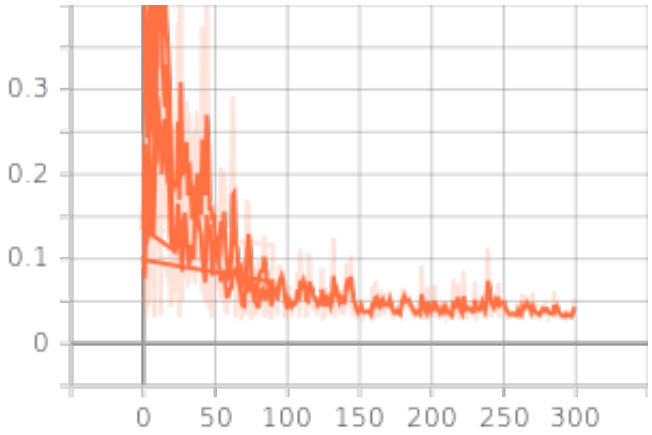

Figure 6: Discriminator Loss for fake data

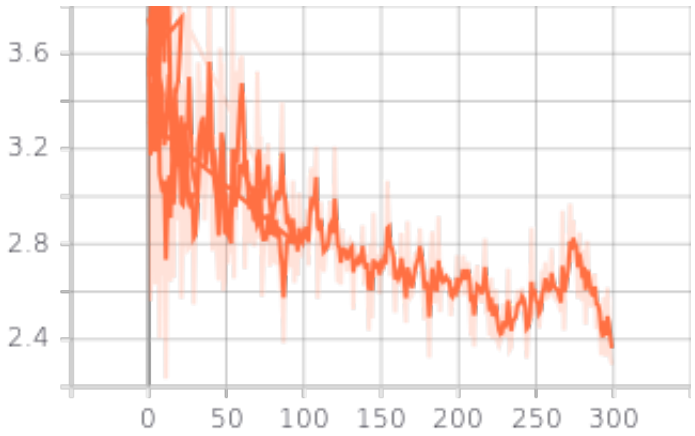

Figure 7: GAN loss

5 Grid of Generated Images

5.1 DCGAN

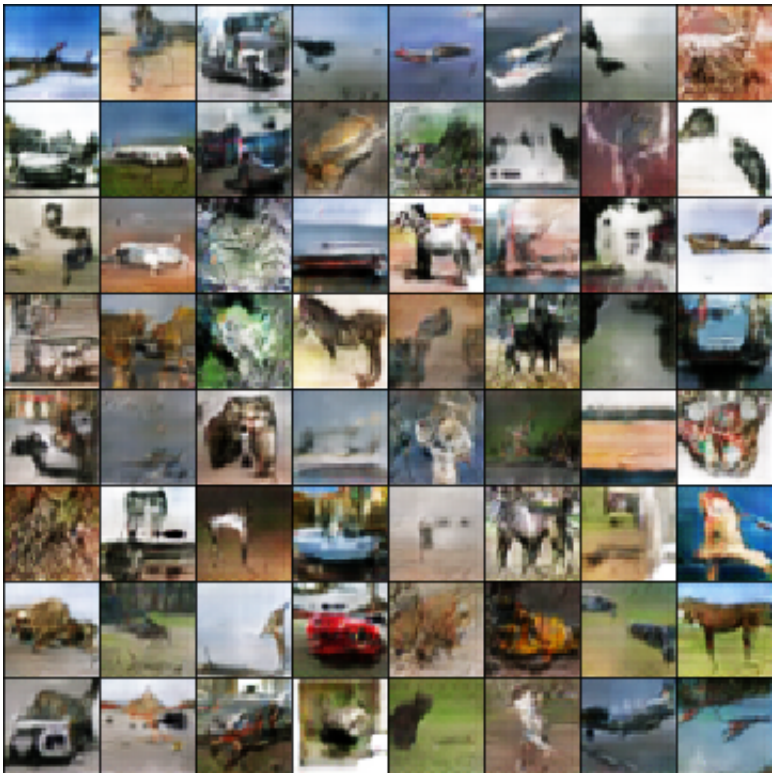

Figure 8: 8*8 Image Grid Generated

5.2 CSAGAN

Figure 9: 8*8 Image Grid Generated

6 Performance Evaluation

6.1 DCGAN

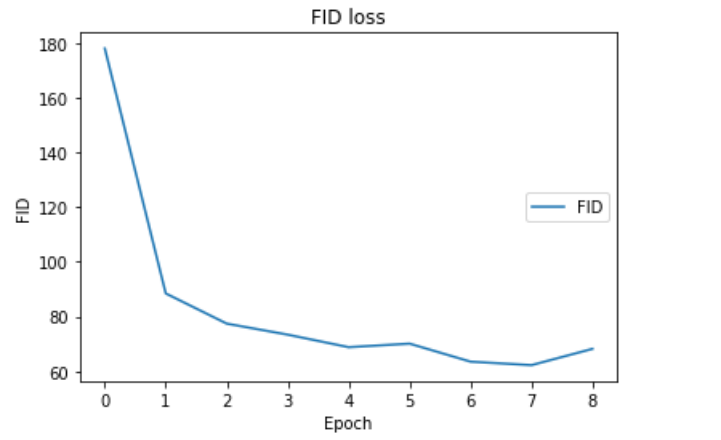

Figure 10: FID for DCGAN

6.2 CSAGAN

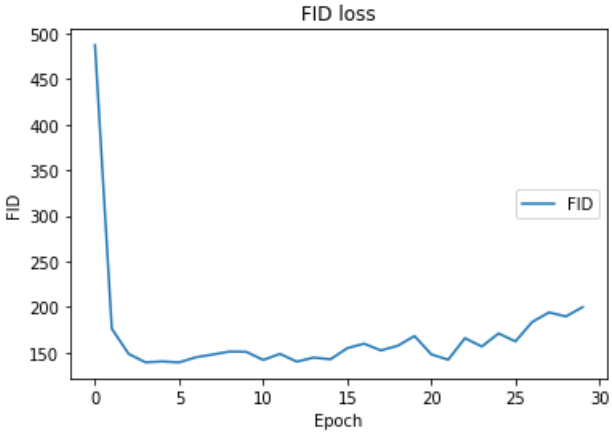

Figure 11: FID for CSAGAN

DCGAN	65.96
SAGAN	143.653

7 References

DCGAN implementation ention GaSelf Att Self Attention Gan WGAN WGAN