Generative Adverserial networks

Badrinath Singhal (23302), Gaurav Talekar (21030)

October 2023

We trained a DC GAN for image size $64 \times 64 \times 3$ using CNN based model. The discriminator used consist of 5 convolutional layers where each layer is followed by batchnormalization and Leaky ReLU. The final layer of discriminator had 1 neuron as output which will act as a classification output. We used sigmoid on that layer to make the output between 0 and 1. Similarly we used 5 conv transpose layer which helps in upsampling where each layer is followed by batchnorm and relu.

We used latent dimensions as 30 so generator model maps 30 dimensional vector to a $64 \times 64 \times 3$ matrix. We trained both the network using Adam optimizer with learning rate of $1e-4$.

To conduct experiments on various number of generator iterations compared to discriminator iterations, we reduced learning rate of discriminator optimizer appropriately compared to that of generator, so one update of generator is roughly equal to more iteration compared to discriminator. We could also do multiple iterations of generator without changing learning rate of both the optimizer.

The loss graph for training GAN for different number of iterations of generator compared to discriminator can be seen in fig 2. Fig 7a refers to normal DCGAN where both generator and discriminator have same learning rate (i.e. same iterations) whereas fig 7b corresponds to two generator iterations for one discriminator iterations. Similarly fig 7c shows loss for 10 generator iterations for each discriminator iterations and finally fig 7d refers to 15 generator iterations. Their corresponding results are shown in fig 1 where fig 1a, 1b, 1c and 1d shows generated images for $1, 2, 10$ and 15 corresponding generator iterations.

We can see that discriminator completely fits the data within 3 iterations whereas generator takes multiple iterations (5-10 epochs) to generate not garbage values. This shows that training GAN is very sensitive and it is easy to overfit discriminator. But since the nature of GAN is to find a saddle point in the loss curve, overfitting discriminator makes finding saddle point tough and thus the GAN training becomes difficult.

It is also very easy to get mode collapse while training even though generator is able to trick the discriminator. It fails to learn to represent complex real world data distribution and gets stuck with low variety. Overall, it is very tricky to train GAN with variety of issues like generator doesnt not have accurate feedback (because of discraminator poor performance) and loss function cannot represent the reality. If discriminator does good, gradients goes to zero and no feedback to generator etc.

To tackle the difficult nature of GAN training it is important to prevent discriminator from overfitting (although that still is not enough in general), this can be achieved by many techniques. For example, we can have more iterations of generator compared to discriminator, we can reduce the learning rate of discriminator so it would reach its optimal point slower or we can skip discriminator updates for few iterations for every generator iterations. All these methods allow us to prevent discriminator to overfit on the data and give room to generator to train.

The results of GAN can be refered in fig 1, we can see that the generated images for all the experiments were roughly looking like the datasets it was trained on. Although it is clearly observed that GAN fail to understand the global pattern properly and we can see many failure cases where the some pictures have one eye or 3 eyes or without nose etc. This shows that GAN doesn't capture the global semantics of the data which can be further improved upon by experimenting with architecture.

We trained W-GAN on the given animals dataset using class wise condition with the method mentioned in the class. For making sure that the network is Lipschitz continous we used weight clipping by clipping the weights to a small window like $[-0.01, 0.01]$ and train the network with new loss function which is not a saddle point finding problem. Since WGAN does't have adversarial nature it is more stable to train and not face the common issues which is faced during GAN training.

We used a fully convolutional network for generator and discriminator model and trained the model on $64 \times 64 \times 3$ dimensions of images of animal faces. We used 5 convolutional layer in discriminator network where each convolutional layer is followed by ReLU and batch norm layer. Similarly we use convtranspose 2d for generatorn layer for interpolation from latent dimension to image dimension. The model was trained with Adam optimizer and a learning rate of $1e-3$ and WGAN training loss.

We found training WGAN was easier compared to DCGAN as mode collapse was not occuring as often as it was in DCGAN. Although WGAN avoids saddle point finding problem, it is still a challenging job to train WGAN. First of all, weight clipping is a bad way to make sure network is Lipschitz continous. WGAN also suffers from unstable training and slow convergences because of clipping window is too large, WGAN also suffers from vanishing gradients when weight clipping window is too small.

Training Wasserstein GAN

-

Linear Dense Network-First the layers of Generator and Discriminator were chosen to linear dense network with 3 layers. This network was not able to give any similar image to the dataset images. The training part was having the problem with gradients. The batch normalization in first experiment was not there which also was added in next model and the weight clipping was added to the training part too. Now, this model was giving some images similar to the dataset but was not able to create a good Generator. So the number of parameters(layers and neurons) were increased which also helped with the images little bit but even then was not able to create a good generator. Also because of the last layer being linear, it helps from gradient going to zero as maybe the case in sigmoid.

Convolution

Then Convolution Layers were added which drastically increased the quality of generator compared to linear dense network. The network was trained for 100 epochs with batch size of 128 images.

Conditionality and Generation

The conditional part of the model was embedded by a embedding vector of same size as number of classes for each class. This is concatenated to the latent vector from Normal Distribution. In generation the class condition was seen following loosely. As we added only 3 dim to the 97 dim latent, the conditionality condition might have been trained strongly if the dimension of the embedding vector was higher and comparable to latent dim. It can be seen in fig 6

loss

The discriminator loss first decreases and then keeps on about same level. Same for the discriminator too. When disriminator changes drastically so does the generator as can be seen at about 20th and 30th epoch. Can be seen in fig 3

Interpolation

The latent space of the latent vector could be said to be continous as could be seen in the interpolation fig 5

BiGAN is modification of GAN to learn latent representations of the data. Additional encoder is used and trained simultaneously along with Generator which will help us get latent representations

for corresponding data.

$f-divergence$ metric is same as that of KL divergence with additional encoder parameters while minimizing generator's loss. Although there is a slight change in discriminator structure, we append both data and its latent representation before we pass it to discriminator. Loss function below was used

$$\min_{G,E} \max_{D} V(D,G,E)$$

where

$$V(D,G,E) = E_{x\sim p_x}[E_{z\sim p_{E(.|x)}}[logD(x,z)]] + E_{z\sim p_z}[E_{x\sim p_{G(.|x)}}[log(1-D(x,z))]]$$

We used encoder with 5 conv layer where each conv layer is followed by a non linearity (ReLU) and batchnorm except the last layer. Is is a fully convolutional model and hence it can take input of anysize. Similarly generator model is 'inverse' of encoder. Although discriminator network is more deeper than generator and discriminator. BiGAN was trained on image size $64 \times 64 \times 3$ with latent dimension as 100. We trained the network for 150 epochs with adam optimizer and learning rate of $2e-4$. Generated images over different epochs can be seen in fig 7. We can see that BiGAN also doesn't have good global understanding of the image as seen by the animal faces details which also occured in DCGAN as well as WGAN.

Training BiGAN is as difficult as training DC-GAN because of the nature of loss function used in Bi-GAN is same as that of standard GAN which is a saddle point finding problem and hence all the difficulty we face while training GAN we face in training Bi-GAN like mode collapse, unstable training, no convergence etc.

We further trained an SVM classifier on the latent representation to classifications. We also trained a linear classifier to compare the performance. Since latent representation dimension is 100, we got accuracy in linear classifier as 56.13% whereas for SVM classifier we got accuracy of 72.85%. We can see that SVM performs significantly better than that of liner classifier and also shows that latent representations learned in BiGAN have some representations of actual data. BiGAN is able to learn latent representations which was not possible in original GAN.

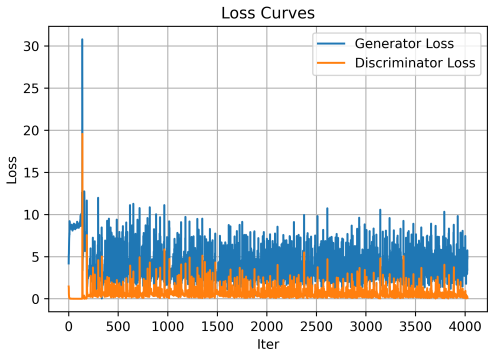
(a) One iteration each

(b) 2 Generator iteration

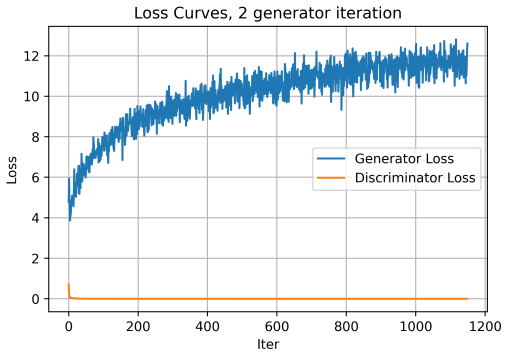
(c) 10 Generator Iteration

(d) 15 Generator Iteration

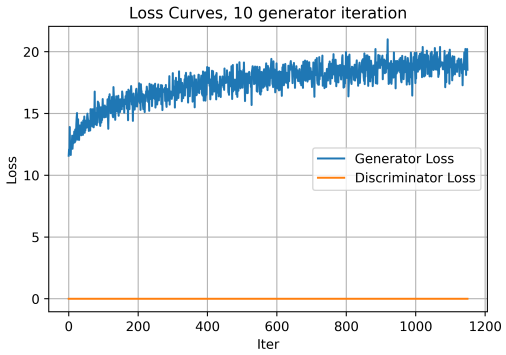
Figure 1: Generated images for DC-GAN with varying generator iterations compared to discriminator



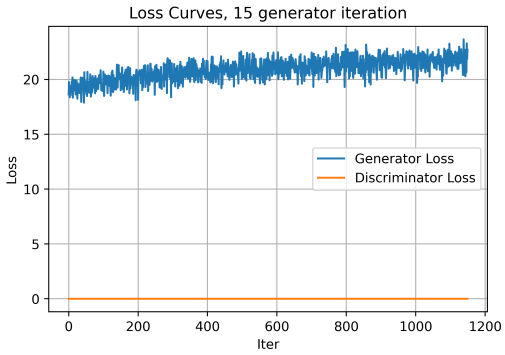
(a) One iteration each



(b) 2 Generator iteration



(c) 10 Generator Iteration



(d) 15 Generator Iteration

Figure 2: Loss curve for DC-GAN with varying generator iterations compared to discriminator

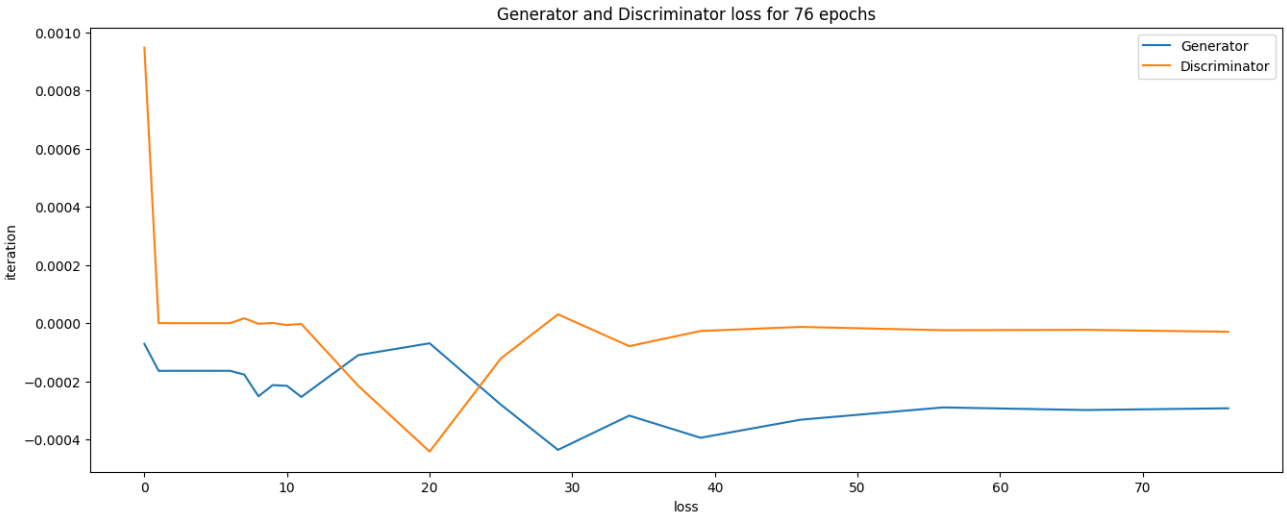


Figure 3: Loss for WGAN training

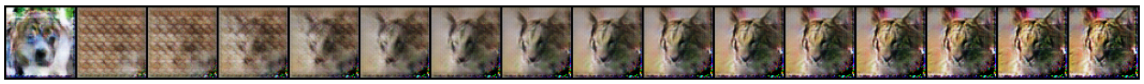
(a) Epoch 50 WGAN images

(b) Epoch 75 WGAN images

(c) Epoch 90 WGAN images

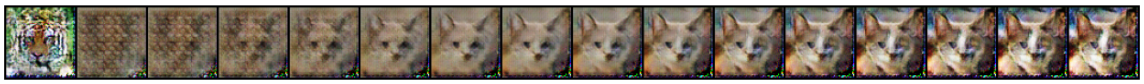
(d) Epoch 100 WGAN images

Figure 4: Reconstructed image for WGAN over different epochs



(a) Interpolation between dog and wild

(b) interpolation between dog and cat



(c) interpolation between wild and cat

Figure 5: Interpolation images for WGAN over different combinations

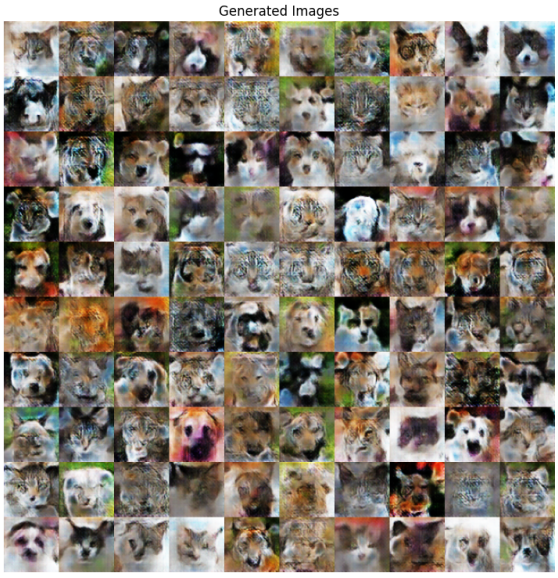
(a) Generated WGAN image for dog

(b) Generated WGAN image for cat

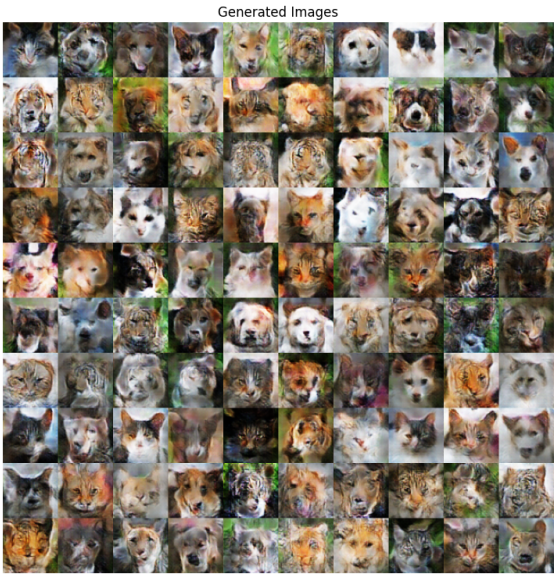
(c) Generated WGAN image for wild

Figure 6: Generated image for WGAN over different epochs

(a) Epoch 60 BiGAN images



(b) Epoch 90 BiGAN images



(c) Epoch 120 BiGAN images

(d) Epoch 150 BiGAN images

Figure 7: Generated image for BiGAN over different epochs