CSI 5340 Assignment 4

Name: Alim Manjiyani

Student Number: 300229095

13th November 2021

1 Aim:

To implement and compare VAE, GAN and WGAN model on MNIST and CIFAR10 dataset, and to investigate their behaviours with respect to complexity and latent space. I have decided to present the report w.r.t the dataset so that it is easy to compare the different models; each section of dataset ends with my observation/conclusion.

2 MNIST

2.1 Visual Auto Encoder (VAE)

2.1.1 Model Setup

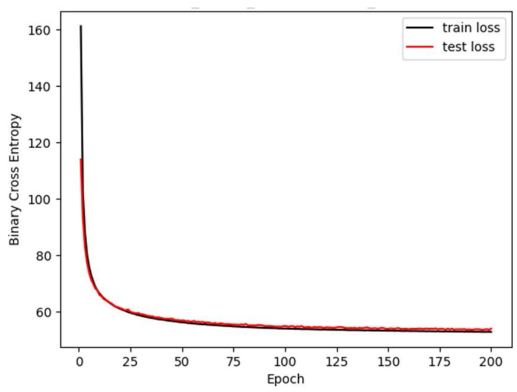
I alter the latent space in the VAE to observe it's behaviour. I compare the two models with 2 and 64 latent vector dimensions respectively. The model setup is described below:

Table 1 VAE Model Setup for MNIST

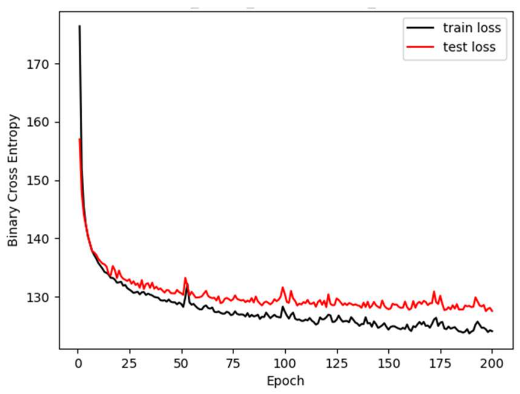
Parameter	*Value*
Encoding Layers	3
Decoding Layers	3
Epochs	200
Latent vectors	2/64
Reconstruction Loss (R_L)	Binary cross-entropy
Total Loss	R_L + KL Divergence

2.1.2 Results:

- Binary Cross Entropy:



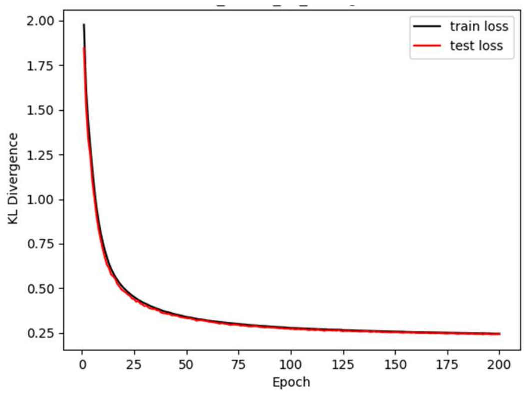
(a) Latent vector dimension: 64



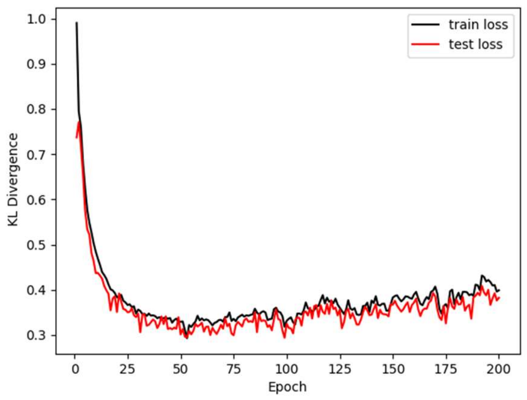
(b) Latent vector dimension: 2

Figure 1: Binary Cross Entropy loss during test and train for VAE models on MNIST

- KL Divergence:



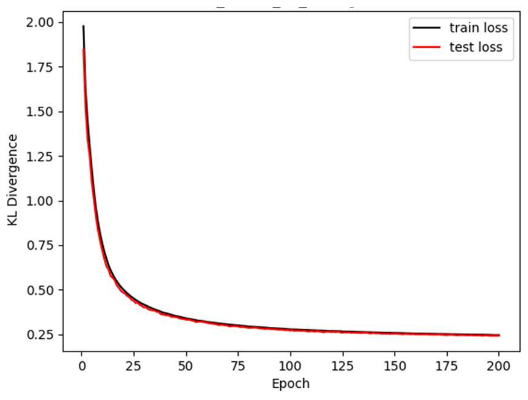
(a) Latent vector dimension: 64



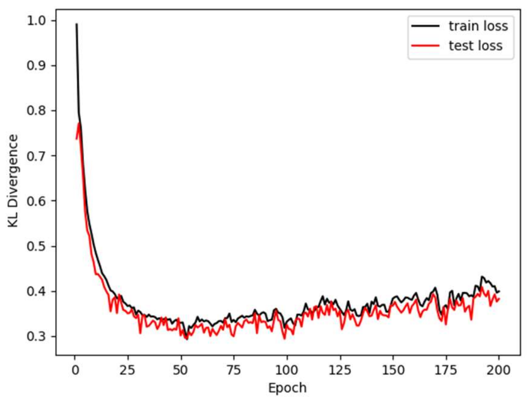
(b) Latent vector dimension: 2

Figure 2: KL Divergence during test and train for VAE models on MNIST

- Total Loss: Reconstruction Loss + KL Divergence:



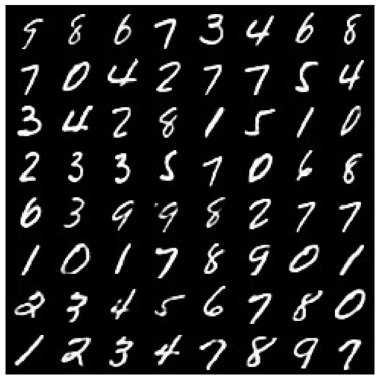
(a) Latent vector dimension: 64



(b) Latent vector dimension: 2

Figure 3: Total loss during test and train for VAE models on MNIST

- Final output:


(a) Latent vector dimension: 64

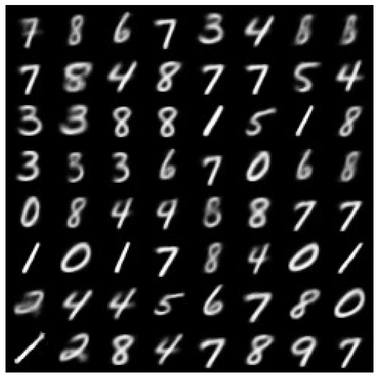

(b) Latent vector dimension: 2

Figure 4: Output after 200 epochs for VAE models on MNIST

2.1.3 Observation:

As expected, performance increases, loss decreases with increase in model complexity (latent vector dimensions). Although performance increases with increase in latent vector dimensions, training time increases as well.

Moreover, convergence of a higher dimension model is smoother than the simpler model, which supports my conclusion.

2.2 Generative Adversarial Network (GAN)

Compared two models with 64 and 128 latent vector dimensions respectively, having 4 linear generator and discriminator layers (1024,512,256,1) each. Each layer has dropout of 0.3 and LeakyReLU activation.

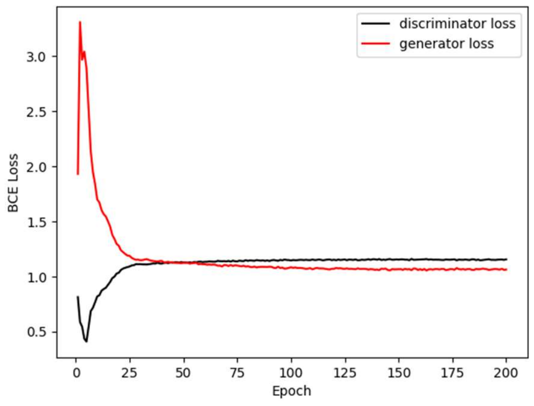
2.2.1 Model Setup:

Table 2 GAN Model Setup for MNIST

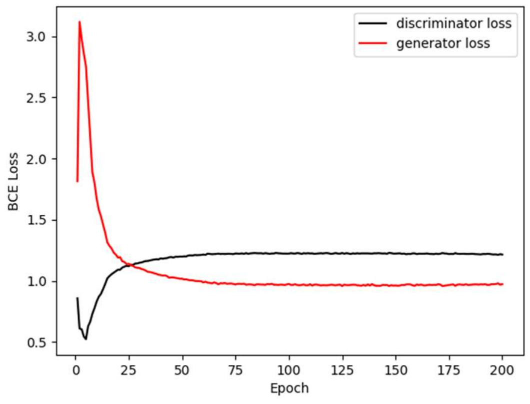
Parameter	*Value*
Generator Layers	4 linear
Discriminator Layers	4 linear
Epochs	200
Activation	LeakyReLU
Dropout	0.3
Latent vector size	64/128
Optimizer	Adam

2.2.2 Results:

- Binary Cross Entropy Loss:



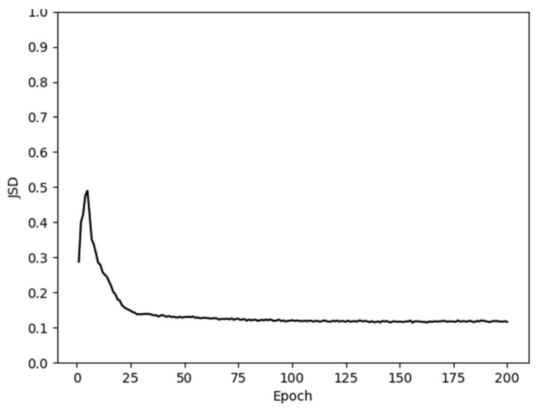
(a) Latent vector dimension: 128



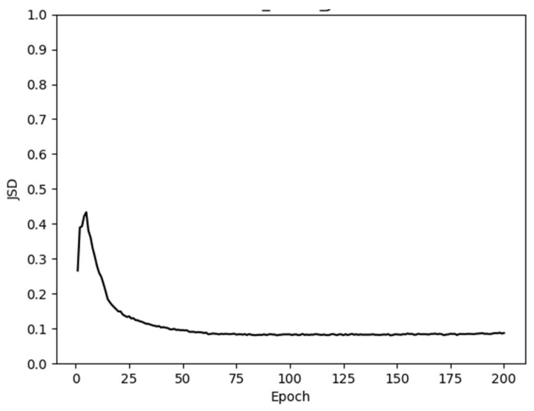
(b) Latent vector dimension: 64

Figure 5: Binary Cross Entropy loss of generator and discriminator for GAN models on MNIST

- Jensen Shannon Divergence (JSD):



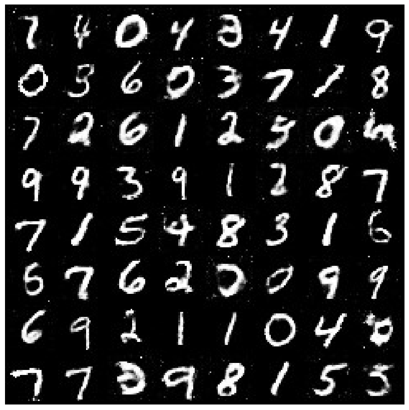
(a) Latent vector dimension: 128



(b) Latent vector dimension: 64

Figure 6: JSD for GAN models on MNIST

- Final Output:


(a) Latent vector dimension: 128

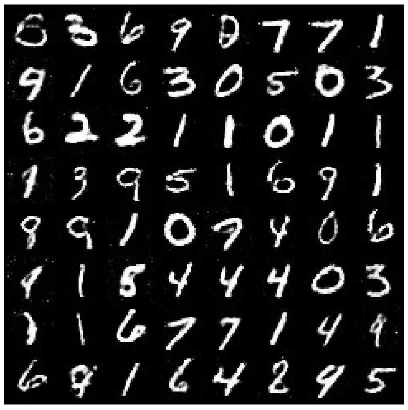

(b) Latent vector dimension: 64

Figure 7: Output after 200 epochs for GAN models on MNIST

2.2.3 Observation:

Although complex model seems to regularize better than the simpler model, JSD for simpler model converges faster which can be accounted to its simplicity. When comparing the output from both the models after 200 epochs, the model with only 64 latent dimension seems to be less noisy/distorted, but the complex model's output is sharper/clear than the simpler model. This is to be expected as higher latent dimensions are prone to noise, and needs more iteration to be noise free.

2.3 WGAN

For WGAN I tried to implement and compare a normal WGAN with DC-WGAN. The only difference between these models is that, while WGAN uses linear layers in generator and critic, DC-WGAN has batch normalized convolutional layers. Both models have 128 latent vector dimensions. Both models have 4 generator neural network layers with ReLU. Critic neural net also has 4 layers with LeakyReLU and dropout. Weight clipping of 0.01 is used along with RMSprop as the optimizer.

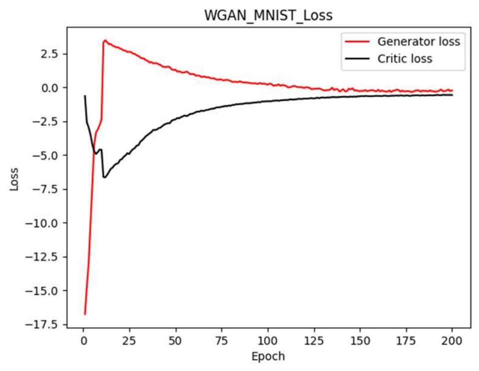
2.3.1 Model Setup:

Table 3 WGAN Model Setup for MNIST

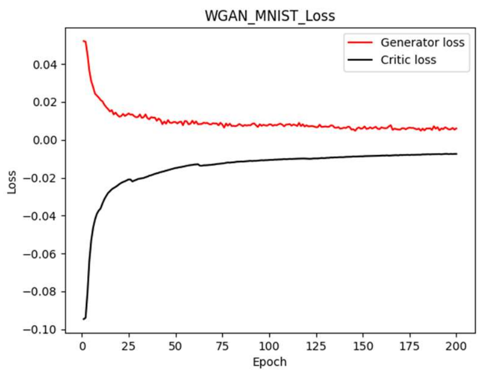
Parameter	*WGAN value*	*DC-WGAN value*
Generator Layers	4 linear	4 convolutional
Critic Layers	4 linear	4 convolutional
Epochs	200	200
Activation	ReLU + LeakyReLU	ReLU + LeakyReLU
Layer settings	Dropout = 0.3	Btach Normalized
Latent vector size	128	128
Optimizer	RMSprop	RMSprop

2.3.2 Results:

- Loss:



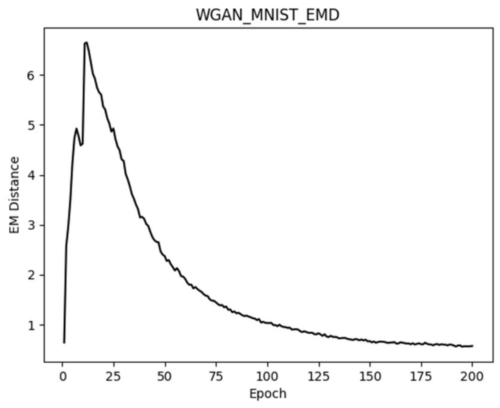
(a) Linear layers



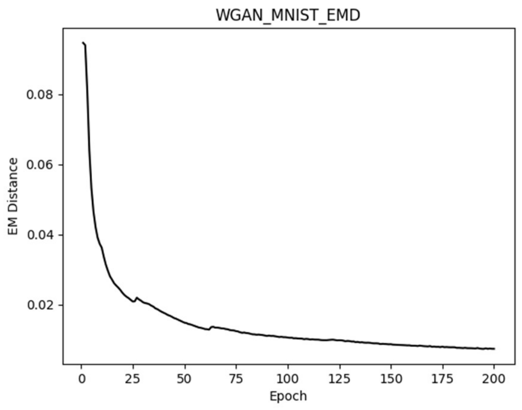
(b) Convolutional layers

Figure 8: Generator and critic loss for WGAN models on MNIST

- Earth Mover's Distance (EMD):



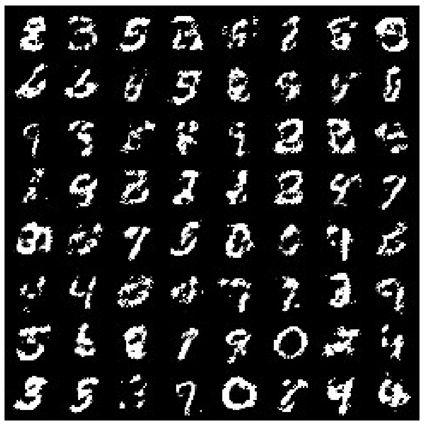
(a) Linear layers



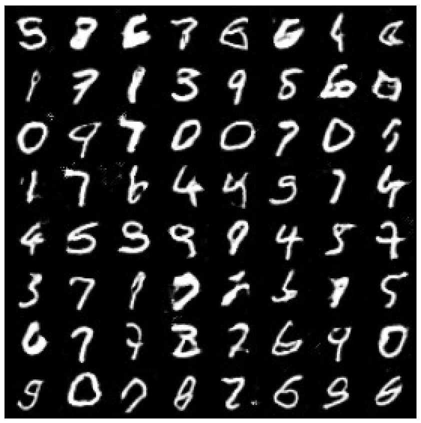
(b) Convolutional layers

Figure 9: EMD for WGAN models on MNIST

- Final Output:



(a) Linear layers



(b) Convolutional layers

Figure 10: EMD for WGAN models on MNIST

Observation:

DC-WGAN outperforms the WGAN. The output image of DC-WGAN is much clear/noise free when compared to normal WGAN. This result can be verified from Figure 9, which depicts the EMD for both the models.

The generator and critic loss for linear model converge faster than the convolutional model, which infers that convolutional model has a chance to improve it's performance in the given setting if iterated more, which doesn't seems to be the case for the simple linear model.

3 CIFAR 10

3.1 VAE

As we know from MNIST that complex VAE gives better results, I wanted to try something else. Observing the positive result of convolutional layers in WGAN on MNIST, I tried to change the linear layers of encoder and decoder to transposed-convolution layers. Both the models have 4 layers and have ReLU activation and batch normalization. This time I have used Mean Squared Error to compute the reconstruction loss which, when combined with KL Divergence gives the total loss.

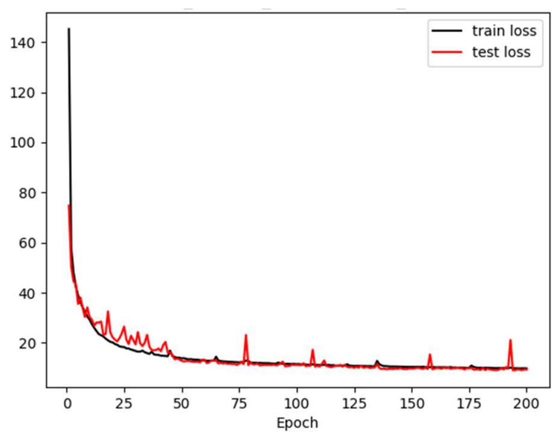
3.1.1 Model Setup:

Table 4 VAE Model Setup for CIFAR 10

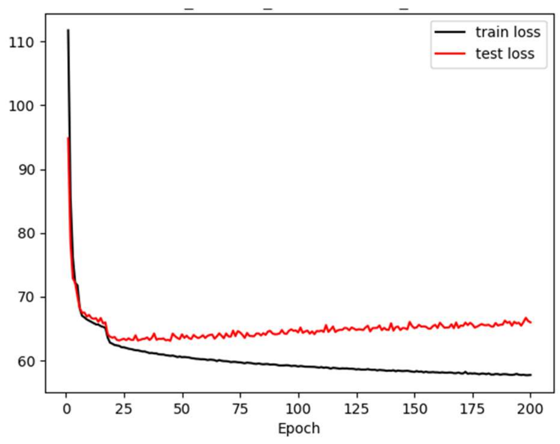
Parameter	*Value*
Encoding Layers	4
Decoding Layers	4
Epochs	200
Latent vector size	512
Activation	ReLU
Layer type	Linear/Convolutional
Optimizer	Adam
Reconstruction Loss (R_L)	Mean Squared Error (MSE)
Total Loss	R_L + KL Divergence

3.1.2 Results:

- Reconstruction (MSE) Loss:



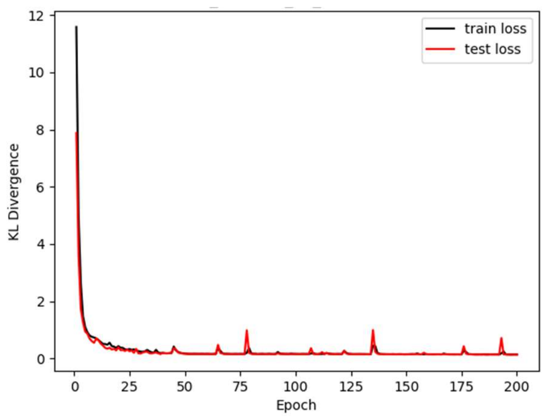
(a) Linear Layers



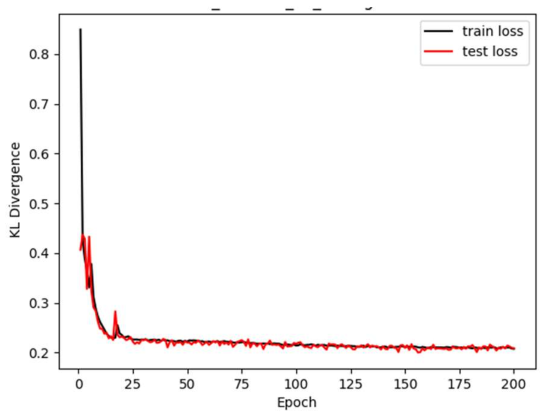
(b) Convolutional Layers

Figure 11: Reconstruction loss during test and train for VAE models on CIFAR 10

- KL Divergence:



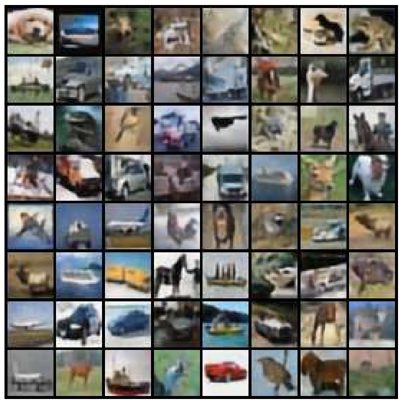
(a) Linear Layers



(b) Convolutional Layers

Figure 12: KL Divergence for VAE models on CIFAR 10

- Final Output:


(a) *Linear Layers*

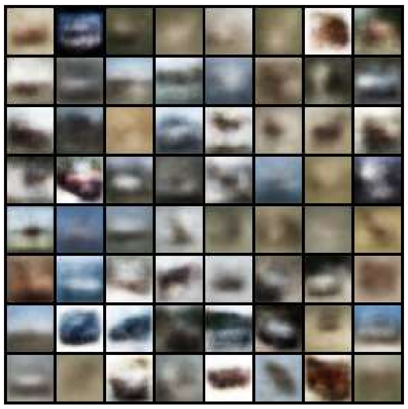

(b) *Convolutional Layers*

Figure 13: Output after 200 epochs for VAE models on CIFAR 10

Observation:

It is clear from the outputs that Linear Layers prevail over convolutional layers in VAE. Although KL Divergence for convolutional layers was less than the linear layers, the reconstruction loss was very high. The poor performance of convolutional model can be attributed to the disability of the model to decode/reconstruct the images properly.

3.2 GAN

Instead of using the simple GAN, I used DC-GAN, which is the same as GAN but has convolutional layers instead of linear layers. I tried to observe the effect of latent vector dimension in this model.

The generator and discriminator have 4 layers each. While the discriminator uses LeakyReLU activation, generator uses ReLU activation. All layers are batch normalized and train oer 200 epoches.

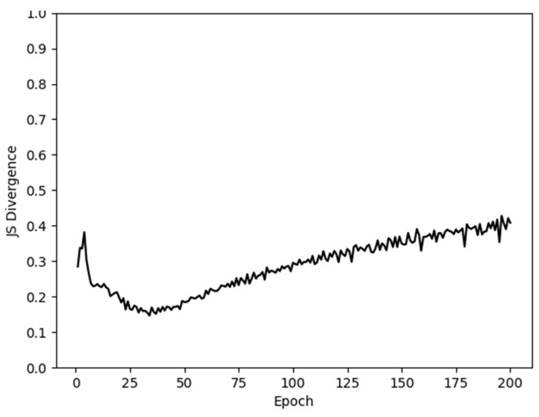
3.2.1 Model Setup:

Table 5 GAN Model Setup for CIFAR 10

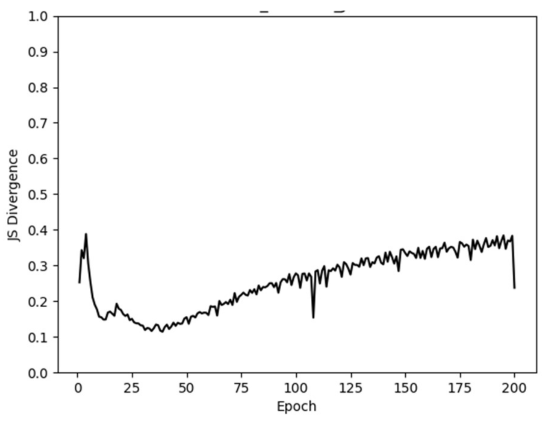
Parameter	*Value*
Generator Layers	4
Discriminator Layers	4
Epochs	200
Generator Activation	ReLU
Discriminator Activation	LeakyReLU
Loss function	Binary Cross Entropy
Latent vector size	64/128
Optimizer	Adam

3.2.2 Results:

- Jensen-Shannon Divergence (JSD):



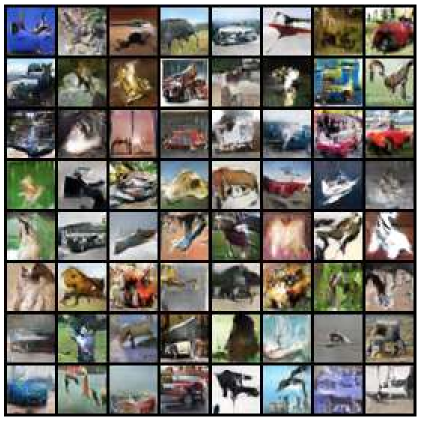
(a) Latent vector dimension: 128



(b) Latent vector dimension: 64

Figure 14: JSD for GAN models on CIFAR 10

- Final Output:


(a) Latent vector dimension: 128

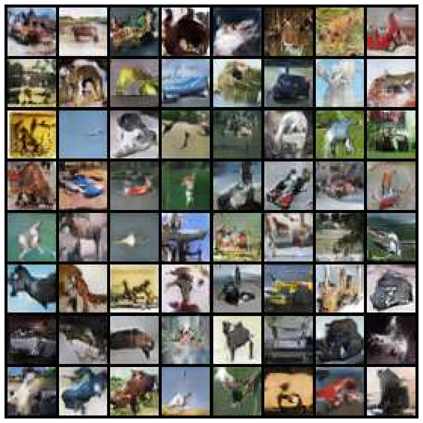

(b) Latent vector dimension: 64

Figure 15: Output after 200 epochs for GAN models on CIFAR 10

3.2.3 Observations:

Replacing the linear layers with convolutional layers doesn't yield better results unlike the case for WGANS. Irrespective of the latent vector dimensions, JSD seems to grow over each epoch after taking a dip on the initial epochs, which states that this model is no good for CIFAR10 datasets. Although the result on CIFAR10 is poor, DC-GANs are said u yield better result on MNIST datasets, which is quite interesting observation.

3.3 WGAN

I decided to use convolutional layers instead of linear layers (DC-WGAN) due to its better performance in MNIST. Both generator and critic have 4 layers with batch normalization. Similar to the model on MNIST, this model uses ReLU for generator and Leaky ReLU for critic. The weight clipping is 0.01.

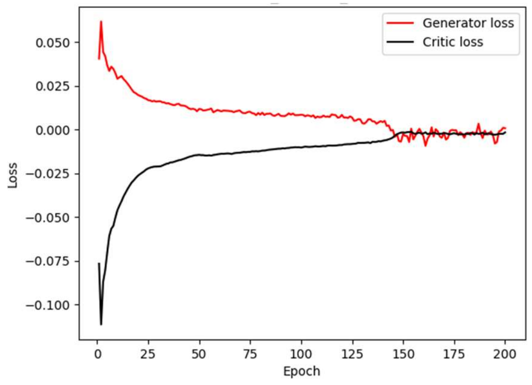
3.3.1 Model Setup:

Table 2 WGAN Model Setup for CIFAR 10

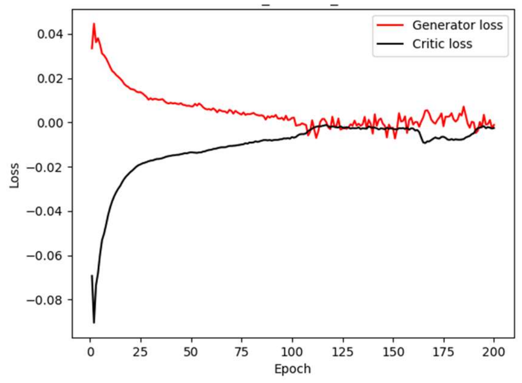
Parameter	*Value*
Generator Layers	4
Critic Layers	4
Epochs	200
Generator Activation	ReLU
Critic Activation	Leaky ReLU
Weight Clipping	0.01
Latent vector size	64/128
Optimizer	RMSprop

3.3.2 Results:

- Generator Loss:



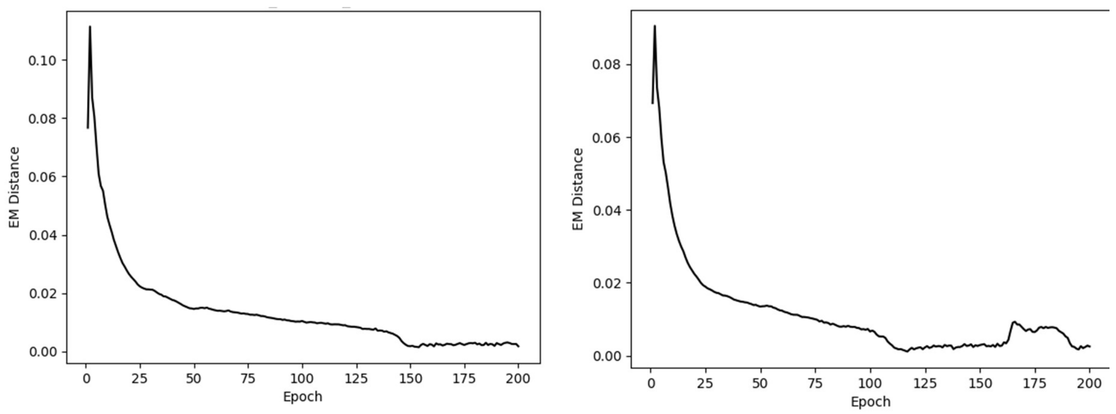
(a) Latent Vector Dimension: 128



(b) Latent Vector Dimension: 64

Figure 16: Generator Loss for WGAN models on CIFAR 10

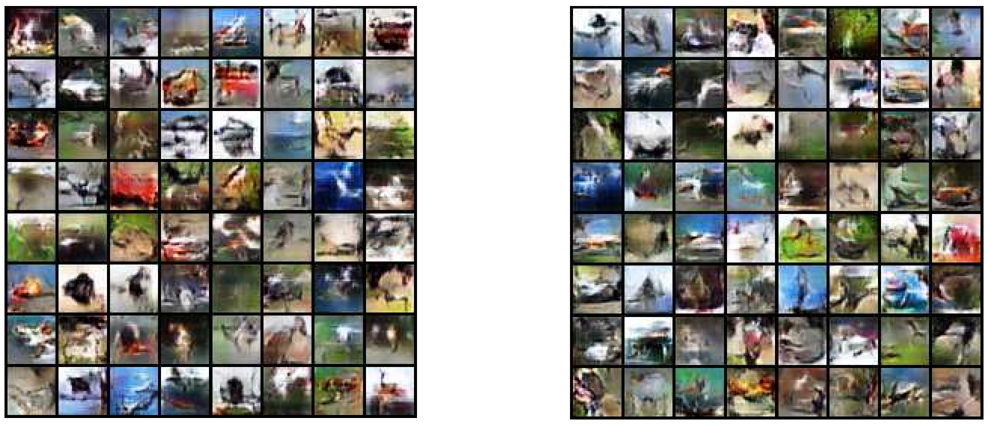
- Earth Mover's Distance (EMD):



(a) Latent Vector Dimension: 128 *(b) Latent Vector Dimension: 64*

Figure 17: EMD for WGAN models on CIFAR 10

- Final Output:



(a) Latent Vector Dimension: 128 *(b) Latent Vector Dimension: 64*

Figure 18: Output after 200 epochs for WGAN models on CIFAR 10

3.3.3 Observation:

As predicted, DC-WGAN shows good result on CIFAR10 dataset. Moreover, it can be seen in Figure 16 and 17 that simpler model converges faster than the complex model. While the complex model seems to converge easily, the simpler model is spiked towards the end but ultimately converges. This abnormality is also seen in Figure 17 which shows the EMD per each epoch. Although there is no clear reason as to why this bump/spike can occur, I think it is because of some technical issue on my laptop and is not related to the code/algorithm.