Deep Learning
CS898BD
Zachary Pearson
D439H842
Assignment 4

Introduction

In this assignment, we compare the performance of Variational Auto-Encoders (VAEs) and Deep Convolutional Generative Adversarial Networks (DCGANs) in image generation using the CIFAR-10 dataset for training.

Methodology

Considering the VAE, we train, simultaneously, an Encoder and Decoder where the encoder takes as input images from the cifar-10 dataset and encodes it to a distribution and the Decoder samples from that distrubution and is trained to reproduce the original input image. The VAE is trained over 100 epochs.

Structural Similarity Index Measure (SSIM) is easily measured for VAE during training to get some measure of how similar the reconstructed images are to the original image given to the encoder. Mean Squared Error will also be used to compare generated images.

For DCGAN, the discriminator and generator are trained seperately. Firstly, the discriminator is fed real images from CIFAR-10 and is batch-trained to recognize them as real. After that, the discriminator is fed images generated by the generator and batch-trained to recognize them as fake. Once the discriminator is trained over both real and fake samples, the generator is is trained to produce more realistic images given the error from the discriminator. This process is repeated for 100 epochs.

SSIM and MSE are also used here to compare the fully trained generators output to the original CIFAR-10.

Architecture

DCGAN: DCGAN architecture was taken from [1]. It is composed of a generator and discriminator which are both combined to form the GAN. The Discriminator and GAN uses binary cross entropy for loss function. The GAN is used to train the generator so the generator does not have an explicit loss function defined for it.

> **Generator:** `dense(4096 neurons) → leakyRELU →`
> `Conv2DT(filters=128,kernel=(4,4),stride=(2,2)) → leakyRelu →`
> `Conv2DT(filters=128,kernel=(4,4),stride=(2,2)) → leakyRelu →`
> `Conv2DT(filters=128,kernel=(4,4),stride=(2,2)) → leakyRelu →`
> `Conv2d(filters=3,kernel=(3,3),activation=tanh)`
>
> **Discriminator:** `Conv2d(filters=64,kernel=(3,3)) → leakyRelu →`
> `Conv2d(filters=128,kernel=(3,3),stride=(2,2)) → leakyRelu →`
> `Conv2d(filters=128,kernel=(3,3),stride=(2,2)) → leakyReu →`
> `Conv2d(filters=256,kernel=(3,3),stride=(2,2)) → leakyReu →`
> `Flatten → dropout(0.4) → dense(1 neuron, activation=sigmoid)`
>
> GAN: Generator → Discriminator

VAE: VAE is made up of both an encoder and decoder. The architecture used here was taken from [2]. A base convolutional block was defined for the encoder to use for its convolutional layers.

Convolutional Block (CB): `Conv2d → BatchNormalization (BN) → Conv2d → BN → M. Pooling`

For the encoder, each convolutional layer has a stride of 1, a kernal size of 3, pooling size of 2, and a pooling stride of 3. The number of filters are given for each layer below (example – CB(128) means a convolutional block with 128 filters)

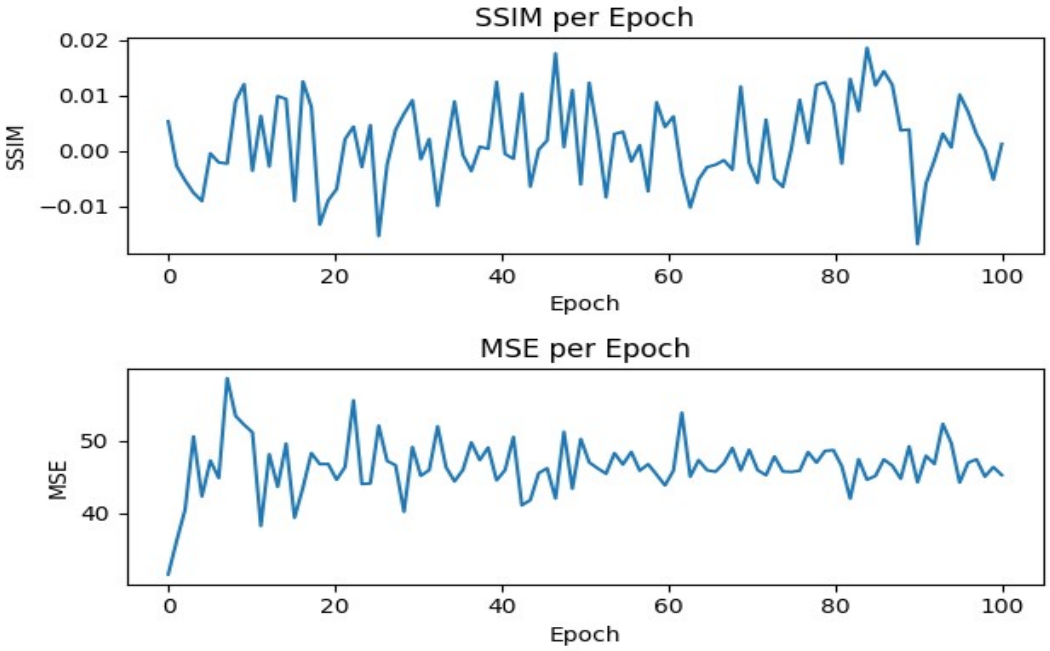
Encoder: `CB(64) → CB(128) → CB(256) → Flatten → Dense(100 neurons) → BN`

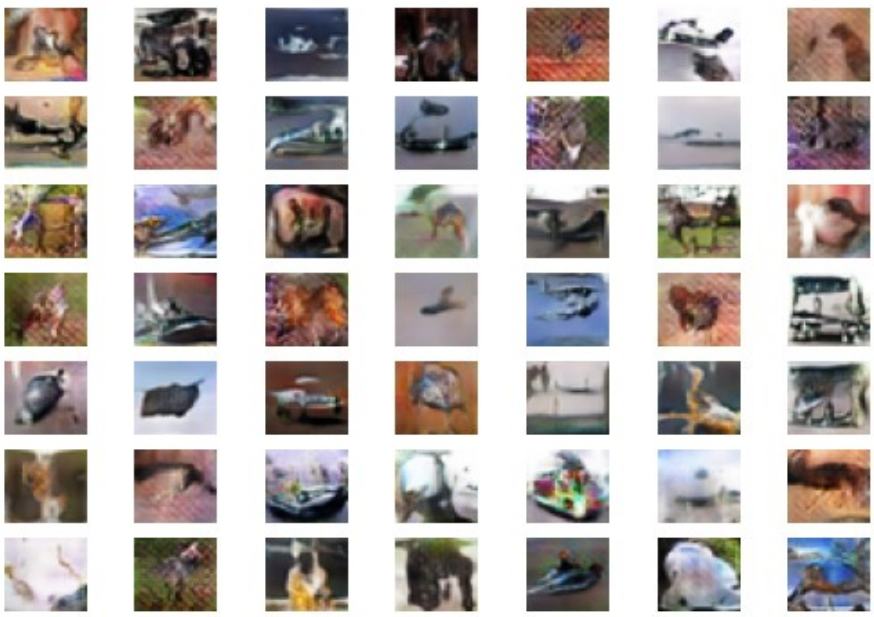
The decoder is composed of several 2D transpose convolutional layers (Conv2dT) with varying numbers of filters separated by Batch Normalization (BN)

Decoder: `Dense(100 neurons) → BN → Dense(1024 neurons) → DB → Dense(4096 neurons) → Conv2dT(256) → BN → Conv2dT(256) → BN → Conv2dT(128) → BN → Conv2dT(128) → BN → Conv2dT(64) → BN → Conv2dT(64) → BN → Conv2dT(3)`

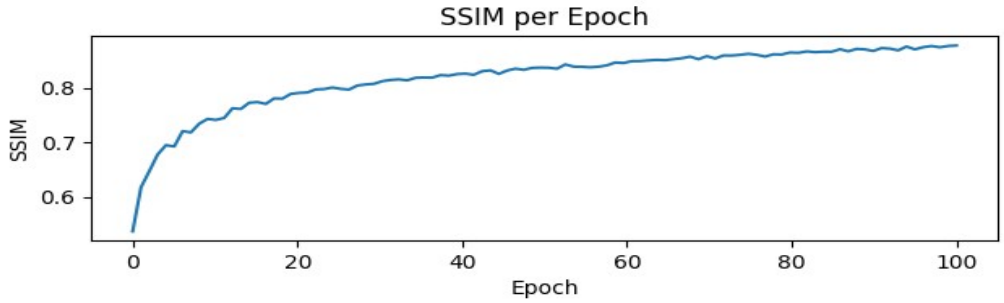
Results

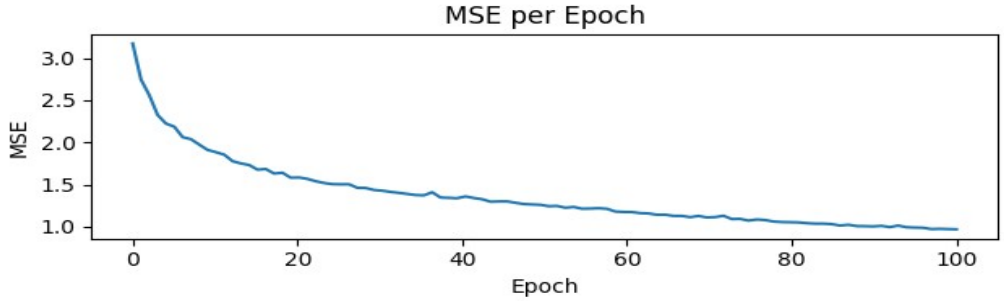
DCGAN:
training time: 3122.810250520706 seconds





VAE:
training_time: 2597.9601390361786 seconds





Conclusion

The main takeaway seems to be the vast difference between their performances as measured by both MSE and SSIM. I think this difference results from what each model is attempting to do. VAE is attempting to learn source images and to be able to replicate those images when given the appropriate distribution. DCGAN, on the otherhand, is attempting to learn to generalize over a selection of real images to produce an image that could trick the discriminator into thinking are real. This latter task does not necessarily rely on reconstructing the original image data, just to capture certain features that the discriminator uses to determine whether an image is real.

References

1. https://www.kaggle.com/code/laszlofazekas/cifar10-dcgan-example
2. https://github.com/arjun-majumdar/Autoencoders_Experiments/blob/master/Variational_AutoEncoder_CIFAR10_TF2.ipynb
3. https://www.kaggle.com/code/alsaniipe/image-similarity-index-ssim-analysis