Comparison Between Generative Models for Reproducing Image Input

Miguel Garcia
University of California, San Diego
mig053@ucsd.edu

Abstract

Here in this paper, I am going to apply three different generative models to two different datasets. The models I will be applying are: Variational Auto-Encoder, Wasserstein Auto-Encoders, and Deep Convolutional Generative Adversarial Networks tensorflow. These models should be able to reproduce input image when well trained. These models vary on the way they train the dataset. DCGAN stabilizes Generative Adversial Networks while VAE operates based on dimensionality while WAE operates in an unsupervised manner. I will tune the hyperparameters for each model so that I can have good samples to compare.

1 Introduction

One thing that has been arising recently is the use of facial recognition and reconstruction technology. It is now being used in smartphones and in many other applications. One could also say there are ethical concerns with this because there has been image reconstruction to spread fake videos but that is one thing that will hopefully be taken care of as this technology arises. Facial and image recognition and reconstruction is being approached in a variety of ways by generative models. These models make use of advanced machine learning techniques to train models and reconstruct images. Comparing the different ways these models perform is essential towards the improvement of these in the future. Understanding the algorithm behind it also contributes towards being able to evolve this arising technology.

2 Method

2.1 Data Sets

In this section, you can see the summary of the datasets and a description of the source code which can be found in each respective source.

The data sets used for this paper are the MNIST [1] and the CelebA [2] dataset.

The MNIST dataset is a database of handwritten digits containing 60,000 examples with 10,000 test examples. It can also be normalized and centered based on fixed-size preference.

The CelebA dataset is a large dataset that contains more than 200,000 celebrity images, with 40 attribute annotations. There is a variety in how these pictures look in terms of pose and background clutter. It also has 40 binary attribute annotations per image

Since I am using Python 3.0, I had to slightly modify the syntax for the code so that it would work using the computer I'm currently using. I downloaded the files from each github respective source.

2.3 Generative Models Used

Variational Auto-Encoder [3] with 60 epochs and dimensionality 20. Which derives a lower bound estimator for a variety of directed graphical models with continuous latent variables. (As described on the source page)

Wasserstein Auto-Encoders [4] which is an unsupervised generative model proposed by Tolstikhin, Bousquet, Gelly, and Schoelkopf.

Deep Convolutional Generative Adversarial Networks tensorflow [5] with input height 28 for the MNIST dataset and 28 output height. Input height for the CelebA model is 108 and it is also cropped before training and testing.

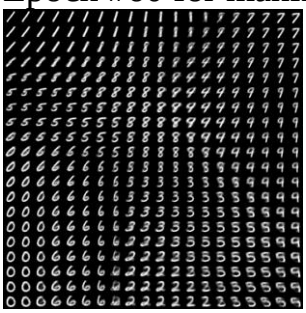
3 Experiment

In this section, I will train each classifier on the data sets and show the results from testing that are outputted to the directory by the program.

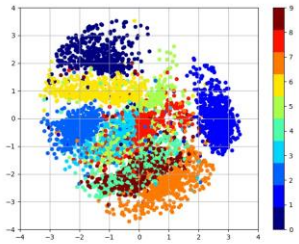
3.1 Variational Auto-Encoder

Input:

Epoch #60 for manifold



Heat Map for epoch #60



3.1.1 Variational Auto-Encoder Performance Analysis

I performed a learned data manifold with 2 dimensions, projecting high dimensional data to a low dimensional manifold. The latent space is Gaussian, which basically produces values of the latent variables z. The probability is plotted for each z value in the results. The output from the command line and input code can be found in the attached file for the results when training and testing to generate the outputted images. You can find more details on the hyperparameters and the source code from the cited website.

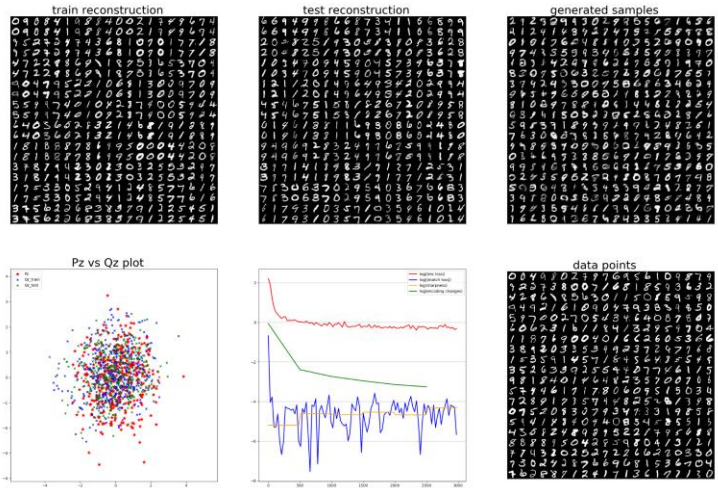
The way VAE works can be described with layers. It starts with vectors as inputs, then a network is trained to change the mean squared error in order to produce the target image. It is possible to make a comparison with the output image, which is not possible using the GAN generative model.

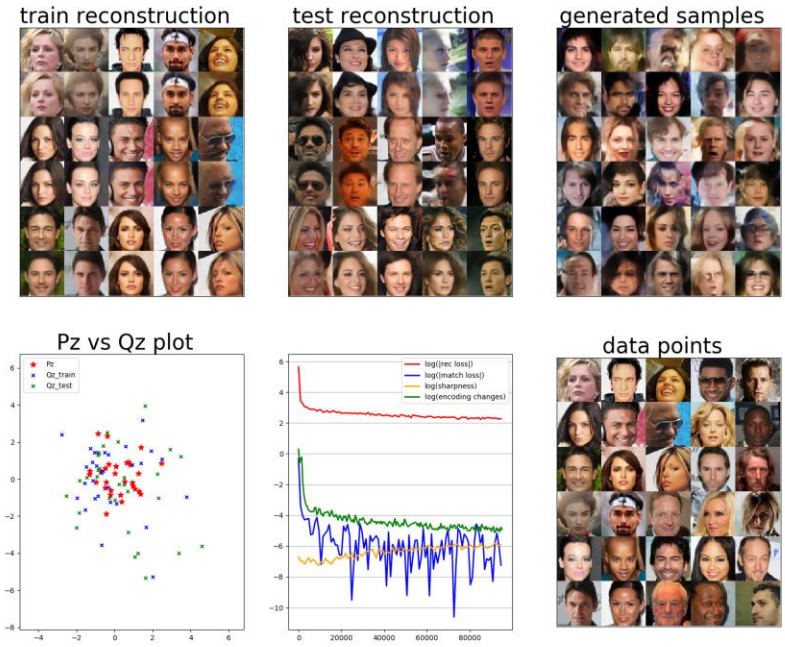
Ideally, I trained the model with 60 epochs to try and get good quality samples, but after a couple epochs, perhaps the 9^{th} one, the results were looking just like the input image. The manifold training works in a different manner than the original one, which is the reason I performed 60 epochs for that case, taking a long time for training.

The performance varies when it comes to the CelebA dataset. You can see the results on the attached file. It performs well after a certain amount of epochs.

3.2 Wasserstein Auto-Encoders

Results after the 5 epochs:





3.2.1 Wasserstein Auto-Encoders Classifier Performance Analysis

I included the training results in an attached file along with the testing results. Above you can see the output given by the algorithm after performing training and testing. This algorithm minimizes the penalized form of the distance between model and target distribution. One way to compare this with VAE is in regularization. WAE performs encoded training to match the previous one. More details can be found in the documentation from their website. WAE also seemed to generate higher quality samples than VAE did.

Here I attach the algorithms use by this generative model, this attachment can be found with the following source [6]

Algorithm 1 Wasserstein Auto-Encoder with GAN-based penalty (WAE-GAN).

Require: Regularization coefficient $\lambda > 0$.
Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.
while (ϕ, θ) not converged **do**
Sample $\{x_1, \dots, x_n\}$ from the training set
Sample $\{z_1, \dots, z_n\}$ from the prior P_Z
Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$
Update D_γ by ascending:

$$\frac{\lambda}{n} \sum_{i=1}^{n} \log D_\gamma(z_i) + \log\big(1 - D_\gamma(\tilde{z}_i)\big)$$

Update Q_ϕ and G_θ by descending:

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) - \lambda \cdot \log D_\gamma(\tilde{z}_i)$$

end while

Algorithm 2 Wasserstein Auto-Encoder with MMD-based penalty (WAE-MMD).

Require: Regularization coefficient $\lambda > 0$, characteristic positive-definite kernel k.
Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.
while (ϕ, θ) not converged **do**
Sample $\{x_1, \dots, x_n\}$ from the training set
Sample $\{z_1, \dots, z_n\}$ from the prior P_Z
Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$
Update Q_ϕ and G_θ by descending:

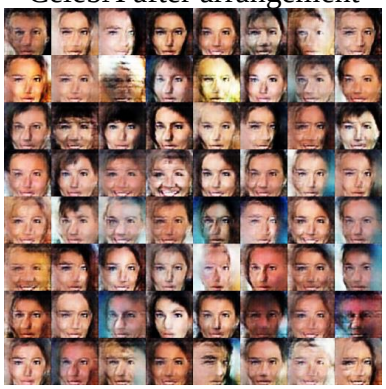
$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(z_\ell, z_j)$$
$$+ \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(\tilde{z}_\ell, \tilde{z}_j) - \frac{2\lambda}{n^2} \sum_{\ell, j} k(z_\ell, \tilde{z}_j)$$

end while

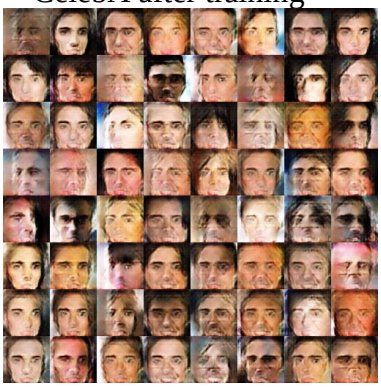
3.3 Deep Convolutional Generative Adversarial Networks tensorflow

MNIST after training

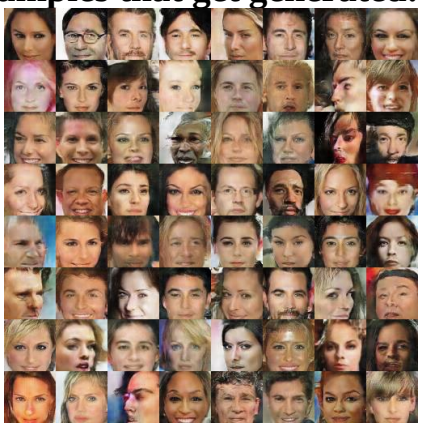
CelebA after arrangement



CelebA after training



After testing these are some of the samples that get generated:



3.3.1 Deep Convolutional Generative Adversarial Networks tensorflow Performance Analysis

I trained the algorithm on 2 different datasets, training took approximately 6 hours for each dataset. After performing the training and testing, some of the samples generated are included above. The rest can be seen on the attached files. The performance of DCGANs is different from the others as it is a form of unsupervised machine learning using a special type of neural networks. The main issue with this algorithm is the amount of time it takes to train for these 2 datasets. Overall it gives performance but it could possibly be more optimal to use WAE since it generates high quality samples as compared to VAE (both take couple of minutes to train).

DCGAN works together with the CNN algorithm to generate images that are validated with CNN. It keeps extracting features with every iteration using probabilistic learning that is included in the CNN. At first the samples produced look “blurry” but as the algorithm keeps training with the samples, the results look more clear and valid. It works in a recursive manner where the convolutions get subsampled, and this process gets repeated over and over until everything can be fully interconnected. One of the main problems with this generative model is the continuity of the latent space. It should not be a problem if the intended purpose when using this algorithm is replicating an image.

4 Conclusion

Running these three generative models helped get a good insight on performing image reconstruction of different types with the use of machine learning techniques. Overall these 3 algorithms I analyzed gave good performance and generated good samples. But comparing them gave me an insight on which one is the most optimal for use. The behavior and implementation of each of these algorithms is different from one another. DCGANs is an unsupervised algorithm which takes a significant amount of time when training, whereas VAE and WAE don’t take as long. Even though VAE uses a form of Gaussian latent space, it was still not generating high quality samples like WAE’s encoded training algorithm generated. VAE can still be used for different types of data, could be sequential, discrete, unlabeled, etc. It will be an optimal choice to use WAE due to the samples generated and the amount of time it takes for training which is not as much as compared to the unsupervised algorithm DCGANs. DCGANs has its cons but if the only thing that you want to get out of it is an accurate replication of any given input image, then this model will work just fine for that intended purpose. All the details about the console output, training and testing samples generated can be seen in the files I attached along with this report.

5 References

[1] Yann LeCun, Corinna Cortes, Christopher J.C. Burges (1998). The MNIST Database of handwritten digits. [http://yann.lecun.com/exdb/mnist/]. Courant Institute, NYU; Google Labs, NY; Microsoft Research, Redmond.

[2] Ziwei Liu and Ping Luo and Xiaogang Wang and Xiaoou Tang (2015). Deep Learning Face Attributes in the Wild. [http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html]. Proceedings of International Conference on Computer Vision (ICCV).

[3] Hwalsuklee, Diederik P. Kingma, Max Welling (2017). Variational Auto-Encoder for MNIST [https://github.com/hwalsuklee/tensorflow-mnist-VAE]. Universiteit van Amsterdam

[4] Tolstikhin, Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, Bernhard Schoelkopf (2017). Wasserstein Auto-Encoders [https://github.com/tolstikhin/wae]. Max Planck Institute for Intelligent Systems.

[5] Alec Radford, Luke Metz, Soumith Chintala (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [https://github.com/carpedm20/DCGAN-tensorflow]. Indico and Facebook AI research

[6] Alec Radford, Luke Metz, Soumith Chintala (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks[https://arxiv.org/pdf/1511.06434.pdf] Indico and Facebook AI research