Generative Adversarial Networks Using Probabilistic PCA

School Of Engineering And Applied Science, Ahmedabad University
Apexa,Parth Satodia,Pinak Divecha,Rushita Thakkar,Vatsal Patel
Group-7 (5 x 5)

***Abstract*—.This paper shows our progress in Working on Generative Adversarial Network.Generative Adversarial Network is applied on the MNIST data and on Faces data provided in Machine Learning folder.Training a Generative Adversarial Network is a really difficult task and the computation Complexity is very high so in order to reduce the computation Complexity,Probabilistic Principle Components of the Images is used. We feed them into the Generative Model and trained the Discriminator Model to discriminate the probabilistic Principle Components of the real data and fake data.**

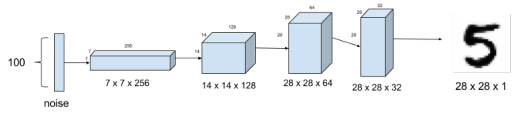
***Index Terms*—Generative Model, Discriminator Model,Adversarial Training,epoch,iterations,Zero-sum game.**

I. Introduction

Generative Adversarial Network is based on semi-supervised learning.Here the generator,generates images(supervised) according to the weights generated by the discriminator where the discriminator is fully trained classifier(Supervised Learning) We apply the concepts of Deep Neural Network(DNN), probabilistic Principle Component Analysis(PPCA), Convolution Neural Network(CNN) on hand written numeric data set and front face data set.As training using the Images could be costly,Probabilistic Principle Component are used so that missing/corrupted entries can be recovered as well as dimensionality reduction can be ensured.

II. Generator Network

The generator basically synthesizes fake images. The fake image is generated from a 100-dimensional noise using the inverse of convolution, called transposed convolution. In between layers, batch normalization stabilizes learning.ReLus are used for activation in each layer. Further generator receives new weights and biases from discriminator and according to that it generates new images and it goes on till it we get the nearly same image as training data image.

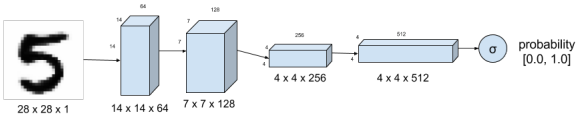


In above image we can see that for given noisy image we are increasing the dimension and reducing number of features in different layers of CNN and relu as an activation function.

III. Discriminator Network

A discriminator that tells how real an image is, is basically a deep Convolution Neural Network (CNN). For MNIST Dataset, the input is an image (28 pixel x 28 pixel x 1 channel). The sigmoid output is a scalar value of the probability of how real the image is. The difference from a typical CNN is the absence of max-pooling in between layers. Instead, a strided convolution is used for down sampling. In other words, Discriminator basically receives image or PC from generator and compare it with image or PC and accordingly gives appropriate weight and biases to generator.

Mathematically, Discriminator tries to learn the mappings directly from space of inputs(Supervised Learning)i.e. p(y—x) Where as the Generator tries to model p(x—y) and p(y)(Unsupervised Learning) based on y = 0 or 1.



In above image we can see that for generated image we are decereasing the dimension and increasing number of features in different layers of CNN and relu as an activation function.

IV. Adversarial Traning

The two network models described above are competing with each other i.e. Generator generates image and discriminator gives out the probability depending on how real the image is and then giving the generator weights and biases.The training of the generative model and Discriminator Model occurs alternatively until the Generator generates images similar to real world data. So both these players are playing a zero sum game at the end the Generative Model generates PC or images that discriminator fails to identify as fake ones(Nash Equilibrium).The discriminator provides the probability of 0.5 for each image.

V. Data Sets

MNIST data set: 60,000 training examples.10,000 testing data Hand Written Neumeric images of fixed-size. i.e. 28X28 grayscale images.
Faces data(100 images as given in Machine Learning Folder) set which is preprocessed and converted into 28 X 28 image grey scale images.

VI. Proposed Model

A. Probabilistic Principle Component Analysis

We will be considering first 20 eigen values The equation: $y = W \times X_n + Tmean$
X is the latent variable of dimension of dimension 20 X 28
Tmean is the mean 28 X 1 (Added to each and every column)
W is the matrix for Probabilistic principle Components. 28 X 20
Here in the model shown below, the generative Adversarial network gives us the matrix W and X .Using this we try to reconstruct the image.

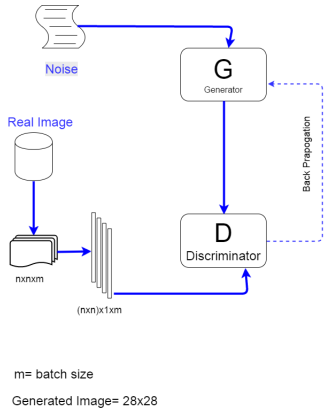


Fig. 1: Flowchart for generative model without PPCA

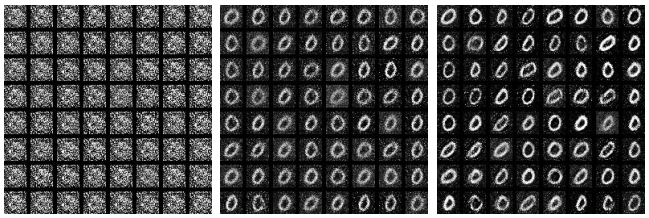


Fig. 2: Output for Epoch 1 Epoch 10 and Epoch 20 for MNIST data set

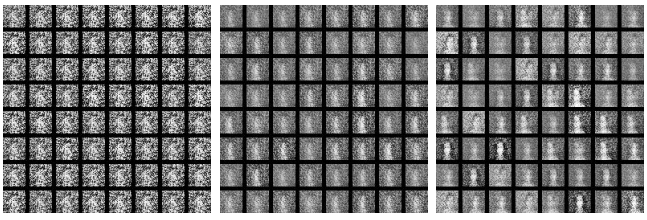


Fig. 3: Output for Epoch 1 Epoch 15 and Epoch 22 for faces Database

Here we are showing the matrix of Principle components and the Mean as images.The Principle Components and the mean vectors are found using the MATLAB code,then stored into image.This is then fed into the python code for ppca GAN

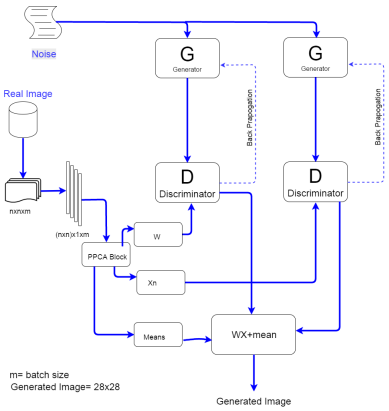


Fig. 4: Flowchart for generative model without PPCA

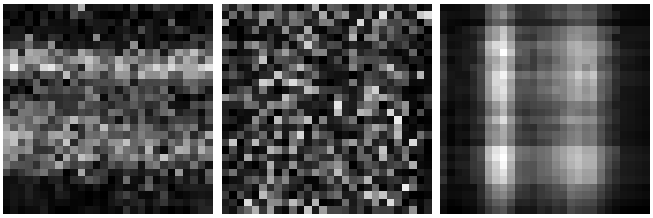


Fig. 5: Generative model output for W, Xn and 'Zero Digit'

VII. Conclusion

So, as seen in the implementation results,the image generated at the end for images (we have taken all the images of digit 0) is fairly good for epoch around 20.Also for Faces data set even though the training data has just 100 images,the output is something similar to a face.But for this,computation cost was very high.It took a long time to generate proper output. When we feed Probabilistic Principle Components,computation cost for training the GAN is significantly reduced but the image is not re constructed properly at the end. (The digit we aim to bring for PPCA at the end is 0 which is some what generated)

Furture Work While feeding Probabilistic Principle Components taking more epochs and iteration can improve the results.This is true even for Faces dataset.

References

[1] Micheal.E.Tipping,Christopher M.Bishop Probibilistic Principle Component Analysis.Journal of Statistical Society.Series B(Statistical Methodology),Vol.61,No 3(1999),611-612.
[2] https://github.com/adeshpande3/Generative-Adversarial-Networks.
[3] https://github.com/ckmarkoh/GAN-tensorflow