Generative models using Probabilistic Principal Component Analysis

Group 5
Akash Soni (1401047), Jay Joshi (1401005), Mihir Gajjar (1401076), Nishi Shah (1401099), Shruti Agrawal (1644007)
$23^{rd} April, 2017$
School of Engineering and Applied Science, Ahmedabad University

***Abstract*—We present new training procedures which we apply to Generative Adversarial Network framework (GAN). We focus on semi supervised learning of GAN and generate principle components from the generative model which was trained using principal components obtained by PPCA. Learning model can be classified into Generative Model G and Discriminative Model D. Training such a network by feeding input images requires high computational and time complexity. Our task aims to present the idea of reduced computational complexity by use of Probabilistic Principle Component Analysis to train the network.**

***Index Terms*—Generative adversarial network, Probabilistic Principle Component Analysis, Deep Convolutional Neural Network, semi-supervised learning**

I. Introduction

Instead of making the Generative Model learn on images and generating an image from the Generative Model, principle components are used to train the generative model.It decreases the computation cost.Two different algorithms are applied on different datasets for comparative analysis.Probabilistic principle component analysis(PPCA) algorithm to fill the missing entries in the Christmas tree Dataset and Principle Component Analysis(PCA) on the MNIST dataset has been applied. The obtained principle components from both algorithms are individually given as an input to the Generative model and a principle component is generated by the model.

II. Generative Model

Generative models is a branch of unsupervised learning techniques in machine learning. It is a model for randomly generating observable data values, typically given some hidden parameters. They are used for artistic applications, image completion and to generate novel content.

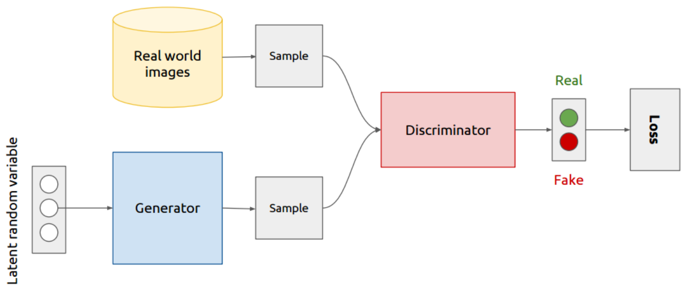

Figure: 1 Representation of Generative Adversarial Model

III. Training a Generative Model

The steps involved in the training are:

- Updating the discriminator to make it better at discriminating between real images and generated ones.
- Updating the generator to make it better at fooling the current discriminator.

We train the generative models to adjust the network parameters so that the generator produces more believable images in the future. Here, the Generative Adversarial Networks approach is used to train the model. In this approach along with the Generator there is a discriminator network that tries to classify if an input image is real or generated. We train the discriminator as a standard classifier to distinguish between the real and the fake images.

- Generator aims to create images that make the discriminator think they are real. In the end, the generator network outputs images that are indistinguishable from real images for the discriminator.Generative algorithms model p(x,y), which can be transformed into p(y|x) by applying Bayes rule and then used for classification by the discriminator.

The training to the discriminator and the generator is given alternatively.

Training the Discriminator:

- Fix generator weights, draw samples(images) from both real world and generated images.
- Train discriminator to distinguish between real world and generated images.

Training the Generator:

- Fix discriminator weights.
- Sample image from generator.
- Backpropagate error through discriminator to update generator weights.

The loss is back propagated to adjust the generator parameters so that the images it generates looks slightly more real or are slightly more confusing for the discriminator. This procedure is repeated until the generator produces an image which the discriminator classifies as real (which may not happen).
Eventually the generator gets so good that it is impossible for the discriminator to tell the difference between real and generated images.

IV. Discriminator

In practise, the discriminator is a deep convolutional neural network.

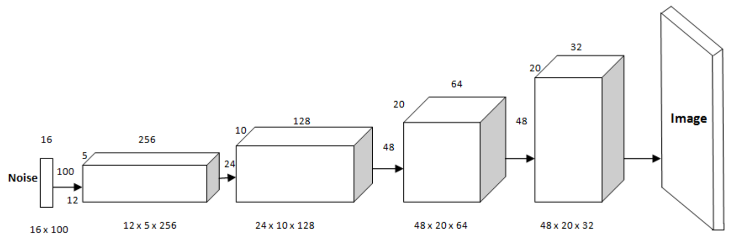

Figure: 2 Discriminator model

This is a classifier function that takes an image as an input and outputs a probability which indicates how real the image is. Each convolution layer uses Rectifier linear unit as an activation function. The convolution operation is performed on the input image, followed by downsampling to reduce the dimensions of the image. The striding convolution method is used to extract the spatial features of the input image in to low dimensional matrix. Dropout of 0.4 is used at each layer which will drop 40 % of neuron weights that helps avoid overfitting.

V. Generator

Generator takes in some kind of noise with a known distribution and transforms it into an image. In practice, the generator is typically a deconvolutional neural network.

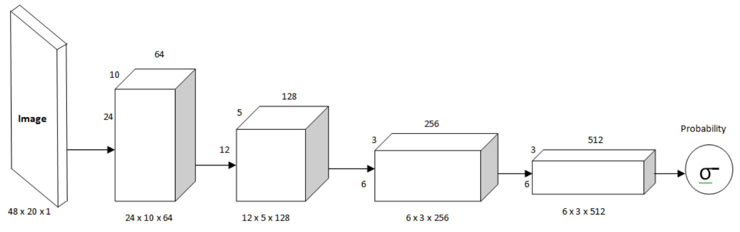

Figure: 3 Generator model

The generator synthesizes fake images from a 100 dimensional noise (uniform distribution between -1.0 to 1.0). Upsampling is used to increase the dimensions of the image instead of deconvolution since it synthesizes more realistic images. Each layer uses Rectifier linear unit as an activation function. The output of the last layer generates an image. Dropout of 0.4 is used at the first layer which will drop 40 % of neuron weights that helps avoid overfitting.

A. *Why convolution in GAN ?*

In images, the pixels in close proximity are more related with each other than with pixels that are a greater distance away. Hence, this gives a better indication of general features that appear in an image by taking advantage of this spatial structure of images.The advantages of adding convolutions and pooling layers to standard neural networks are as follows:

- **Shift Invariance:** Convolutional layers detect a given feature, regardless of its position on an image.
- **Computationally Efficient:** Convolutional networks have fewer parameters involved, making the network more computationally efficient to train. Instead of associating weights to each neuron of fully connected neural network, by applying convolution we only have 1 bias per kernel which drastically reduces the computational complexity involved in further computations of parameter scaling.

.

VI. Proposed Model

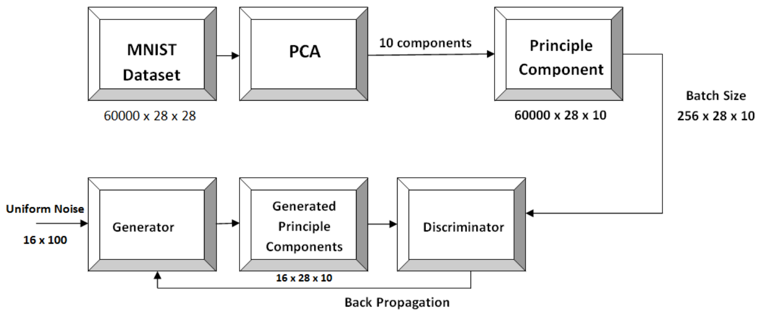

Figure: 4 PCA with Generative Models

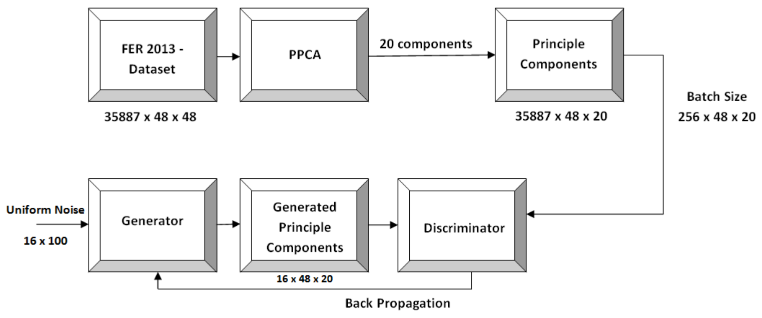

Figure: 5 PPCA with Generative Models

VII. Results

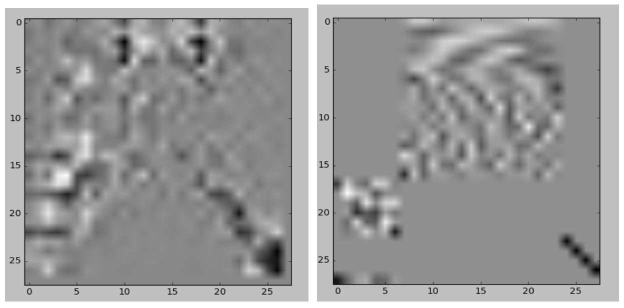

Figure: 6 Principle Component obtained at 50 and 200 iterations respectively using PCA with Generative Model on MNIST dataset

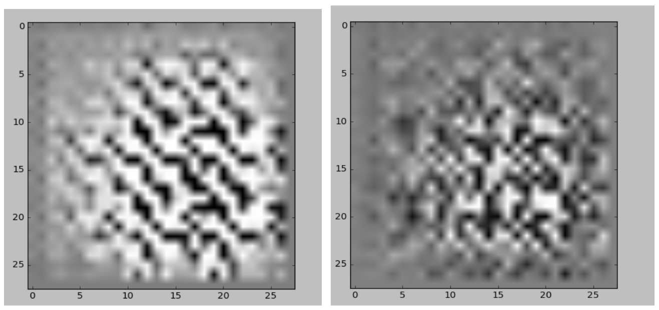

Figure: 7 Principle Component obtained at 50 and 200 iterations respectively using PPCA with Generative Model on Christmas-tree dataset

VIII. Conclusion

We are able to generate principle component using the PPCA algorithm but we are not able to develop a verification technique to verify whether the generated principle component is valid or not. By multiplying the output paramters of the PPCA we can get an image. Hence we tried generating these parameters from the Generative models by training the generative models with the output parameters of the PPCA of the images of a specific category in the data set and using the below equation we multiplied the parameters to generate the image, but we were not able to generate a suitable(real-like) image.

References

[1] Yann LeCun, 'THE MNIST DATABASE of handwritten digits', 2017. [Online]. Available: http://yann.lecun.com/exdb/mnist/ [Accessed: 01 Apr- 2017]

[2] Rowel Atienza, 'GAN by Example using Keras on Tensorflow Backend', 2017. [Online]. Available: https://medium.com/towards-data-science/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0 [Accessed: 05 Apr- 2017]

[3] Ian J. Goodfellow,'Generative Adversarial Networks', 2014. [Online]. Available: https://arxiv.org/abs/1406.2661 [Accessed: 10 Apr - 2017]

[4] Xu-Die Ren, Hao-Nan Guo,'Convolutional Neural Network Based on Principal Component Analysis Initialization for Image Classification',2013. Available: http://ieeexplore.ieee.org/document/7866146/authors [Accessed: 28 Mar - 2017]

[5] Rowel Atienza,'GAN by Example using Keras on Tensorflow Backend', 2016. [Online]. Available:https://medium.com/towards-data-science/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0 [Accessed: 10 Apr - 2017]