Generative Adversarial Networks using Probabilistic Principle Componant Analysis

Group 10
Akshat Doshi, Faheem Samol, Neha Aytoda, Rajvi Prajapati, Ratnesh Shah
School Of Engineering & Applied Sciences, Ahmedabad University.

***Abstract*—In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. We build good image representations by training Generative Adversarial Networks (GANs) and later reusing parts of the generator and discriminator networks as feature extractors for supervised task. We introduce a class of CNNs called Deep Convolutional Generative Adversarial Networks (DCGANs) and they are a strong candidate for unsupervised learning. Generative probabilistic models such as Probabilistic Principle Component Analysis provides a principled way of treating missing information.**

***Keywords*– Generative Adversarial network, Probabilistic Principle Component Analysis, Convolution Neural Network**

I. Introduction

Building a good generative model of natural images has been a fundamental problem within computer vision. However, images are complex and high dimensional, making them hard to model well, despite extensive efforts. Given the difficulties of modeling entire scene at high-resolution, existing approaches instead, generate image patches. At each scale we train, a convolutional network based generative model using the Generative Adversarial Networks (GAN). We have used principal components (PCs) instead of whole image to reduce dimension. We have used a dataset, which has some missing entries to fill those missing values we have used Probabilistic PCA (PPCA) for filling the matrix first and then getting the PCs of the images.

II. Generative Adversarial Networks

The main idea behind a GAN is to have two competing neural network models. One takes noise as input and generates samples (and so is called the **Generator**). The other model (called the **Discriminator**) receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. These two networks play a continuous game and are trained simultaneously so that the competition will drive the generated samples to be indistinguishable from real data.

III. Dataset

We have worked on two datasets. One is using FER2013 which is a facial expression recognition database provided by Kaggle. This dataset consists of 35,887 images of various facial expressions. Each image is of dimension (48x48), after implementing PCs dimensions are reduced to (48x20). This database is comprehensive enough as it contains some of the missing values, thus PPCAs can be incorporated in the model. Another dataset we have worked on is MNIST dataset which is a standard dataset containing 60,000 samples and 10,000 samples for testing.

IV. Our Model

A. Approach1: On FER2013 dataset

We first extract the Principal components of all the images in the dataset using the PPCA method. We use the first 20 PC's. So, now we have reduced the dimension of each image to 48 x 20.

1) Generator: The generator generates fake PPCA's. Generator is given random noise as input of 16 x 100 dimension. We are using deconvolution layers which extracts out the features at each layer and finally generates PC's of dimension 48 x 20 and 16 such samples. ReLU is used as activation function at each layer and sigmoid function is used at the last layer. The model described above can be visualized from the figure below.

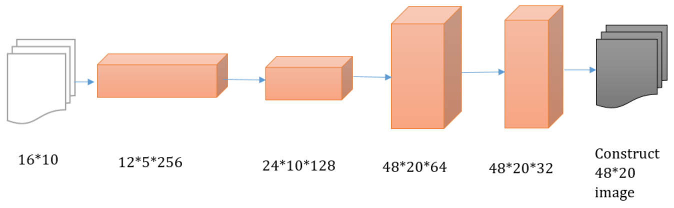

Fig 1: Generator Model

2) Discriminator: The extracted PC's are the given as input to the discriminator in a batch of 256. The fake PC's generated by the generator is given to the discriminator which tells us if the generated PC's are real of fake. It gives an output of 1 if the PC's look like real and 0 otherwise. Since the output of the discriminator is sigmoid, we use binary cross entropy for the loss. The sigmoid output is a scalar value of the probability of how likely the image real. The loss calculated from this probability is then used to re-assign the weights of the generator model. Dropout with probability of 0.4 is used to prevent overfitting. The activation function used in each CNN layer is a leaky ReLU.

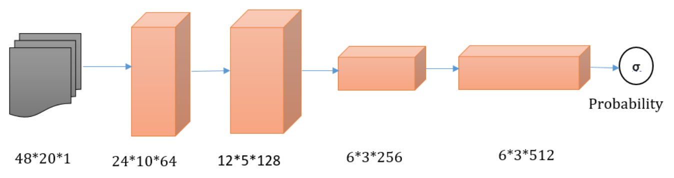

Fig 2: Discriminator Model

The complete model of our approach-1 can be seen from the figure given below.

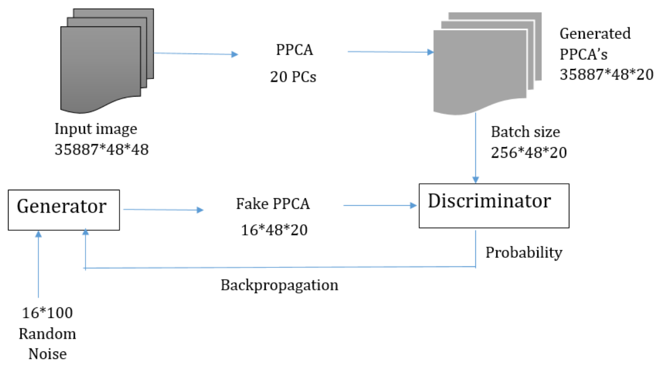

Fig 3: Complete Model

B. Approach2: On MNIST dataset

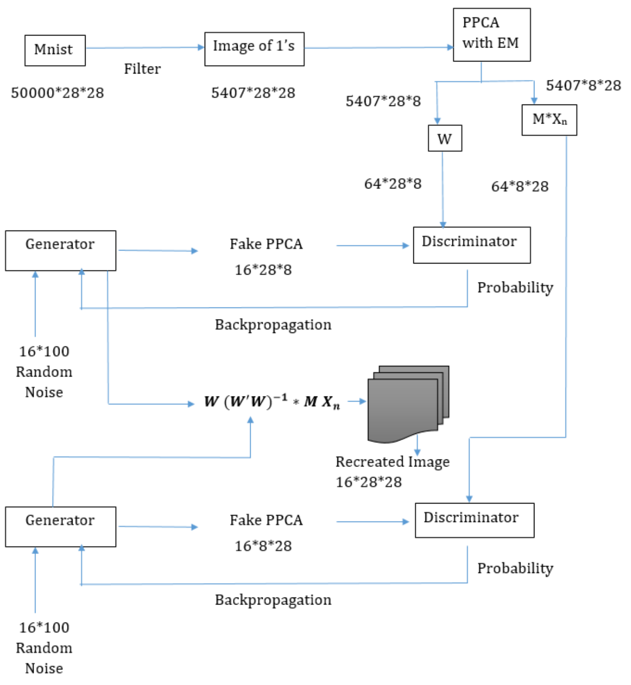

Fig 4: Model

From the PPCA-EM algorithm described in [3] we come to know that we can recreate the image using PC's if we have information of the latent variables M and Xn using the formula $W(WW)^{-1}MXn$. So we first filter the images of MNIST such that we are only left with images of 1's. Then using the PPCA-EM algorithm we extract out first 8 W's i.e. principal components of each image so now (28 x 8) along with their corresponding M x Xn of (8 x 28) dimension.

We are using two GAN models using which we generate the fake PC and also the product of latent variables (M x Xn). These are then used to recreate the image. The complete model of approach-2 can be visualized with the help of above figure.

V. Results

Using the Approach-2 we were able to generate the image which can be seen below which should have been an image similar to that of 1.

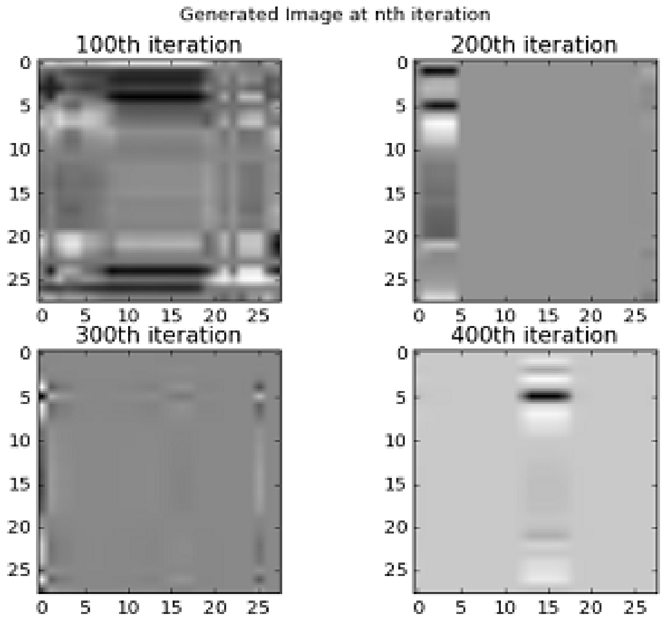

Fig 5: MNIST

Iteration	100	200	300	400
Norm	1204.99	8581.05	856.61	2770006.48

Table-1: Calculated Norm of MNIST data set

Iteration	1000	5000	8000
Norm	50.05	50.06	50.02

Table-2: Calculated Norm of FER data set

VI. Conclusions

We were able to successfully generate PCA but we have no method to verify the correctness. One of the reasons of failure of approach-2 we think is that since we are using the PC's to train the discriminator which used convolution layers to extract the spatial features from the image but when we extract the PC this information itself gets lost due to which it din't come to know how an actual 1 looks like.

References

[1] https://medium.com/towards-data-science/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0
[2] https://github.com/adeshpande3/Generative-Adversarial-Networks/blob/master/Generative
[3] Probabilistic Principal Component Analysis by Michael E. Tipping Christopher M. Bishop, Microsoft research, September 27, 1999.