Generative Adversarial Networks using Probabilistic Principal Component Analysis

Hemang Jethava, Jainil Vachhani, Nand Parikh, Palash Hariyani, Parth Shah

School of Engineering and Applied Science
Ahmedabad University

***Abstract*—Supervised learning with Convolutional Neural Network (CNN) is used extensively in computer vision application.Compared to this unsupervised learning with CNN has attracted comparatively less research interest. This paper aims at bridging the gap between supervised and unsupervised learning using Deep Convolutional Generative Adversarial Network (DC-GAN). A semi-supervised approach where our discriminator is trained to predict fake representations by a given labeled dataset of real or fake representation is approached here. A min-max game approach between generator and discriminator is used. Proposed model instead of directly working on images works on principal components, this novel approach helps in faster computation.**

Keywords: Min-Max game Generator, Discriminator, Adversarial Model, Probabilistic principle component analysis.

I. Introduction

We propose that one way to build good image representations is by training Generative Adversarial Networks (GANs) and later reusing parts of the generator and discriminator networks as feature extractors for supervised tasks.GANs provide an attractive alternative to maximum likelihood techniques.A comprehensive study and detailed explaination on generator, discriminator GANs proposed by[1] have been known to be unstable to train, often resulting in generators that produce nonsensical outputs. A method proposed by [2] amends certain aspects of [1] and has proposed a more stable model for generator and discrminator. Principal components (*PCs*) instead of whole image, is used in our model. [2] has proposed models using images. Our model makes a few changes in the model for accomodating *PCs* as inputs and outputs. We have used a dataset[4] which has some missing entries , hence we have used Probabilistic PCA (PPCA) for first filling the matrix and then getting the *PCs* of the images.

II. Dataset

A standard dataset *MNIST* has been used in [3] to generate new hand written digits. Due to computational limitations, we have used a facial database given by Kaggle[4]. Dataset consists of approx $36,000$ 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.

III. Generative Adversarial Network

This section talks about the model proposed by [2] and sucessfully implemented by [3]. Generator, Discriminator and Adversarial model is dicussed here. The model has made certain amendments in [1] to produce a more stable model.

A. Generator

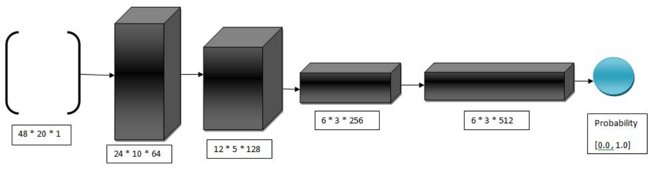

Fig-1:Model for Generator

The generator creates fake images. In Figure 1, the fake image is generated from a 100-dimensional noise (uniform distribution between -1.0 to 1.0). We have avoided the use of fractionally-strided convolution as suggested in DCGAN[1], upsampling between the first three layers is used since it generates more realistic *PCs*[2]. In between layers, batch normalization stabilizes the learning model[2]. The activation function after each layer is a Rectified Linear Unit. The output of the sigmoid at the last layer produces the fake image.

B. Discriminator

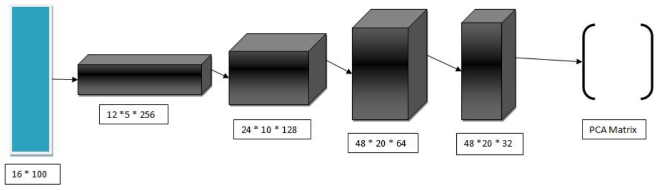

Fig-2:Model for Discriminator

Input for *Kaggle[4]* dataset after getting the *PCs* is (48 pixel x 20 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 downsampling. The activation function used in each CNN layer is a leaky ReLU. A dropout between 0.4 and 0.7 between layers prevent over fitting and memorization.

C. *Adversarial Model*

The adversarial model is just the generator-discriminator stacked together. The generator synthesizes new images which gradually resembles like the real images when generator learns from the feedback of discriminator .

D. *Training*

Training is the hardest part. We determine first if Discriminator model is correct by training it alone with real and fake images. Afterwards, the Discriminator and Adversarial models are trained one after the other.

IV. GM USING PPCA

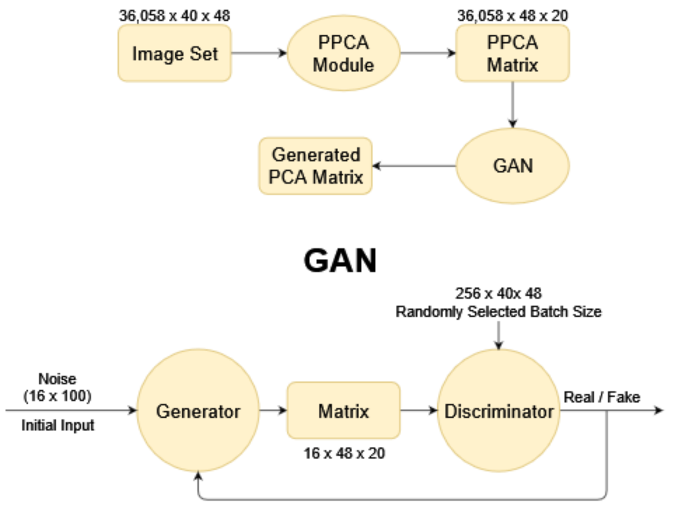

Fig-3:GM using Principal components

The Generative Model (GM) proposed in *section III* is changed to incorporate the *PCs*. This amendment is quite simple instead of giving input as image matrix, first the image matrix is given to PPCA module which generates the *PCs* of the image set, it is then given to the GAN model, where a randomly selected 256 *PCs* are used to train the discriminator thus the dimensions of the training set at a time to discriminator would be(256x48x20). As we need to generate 16 samples an initial noise of dimension(16x100) is the input to generator. Generator generates new *PCs* of dimension(16x48x20) which is feeded to discriminator. Discriminator then tells whether the generated *PCs* are fake or real and the feed back is given to generator.

V. RESULTS

The section talks about the results generated and proof of its correctness.

A. *Generated Results*

No. of Iterations	Error Norm
500	0.92
1000	0.77
5000	0.12
8000	0.088

Table-1 shows the error of the generated principal components, we can see that initially at just 500 iterations we have a huge error, thus we can consider the generated *PCs* that of noise. As iterations increase, error is decreasing, finally we see that at 8000 iterations we have a very lowered error of 8%. Due to computational limitation the results of $10,000^{th}$, step is not shown here but will surely have a lowered error.

B. *Proof of correctness*

Through out the proof the dimensions of principal components is (48x20).

Dataset consists of $36,058$ image's principal components. We have generated 16 principal components at each iteration.

The probability that any random 256 image batch is taken for discriminator can be given as:

$$p_{discrand} = \frac{256}{36058}$$

Similarly from these 256 randomly selected images the probability of getting a (16x48x20) matrix resembling our generated principal components from generator can be given as:

$$p_{genrand} = \frac{16}{256}$$

Let A_{16} is random 16 principal components from 36058 *PCs*.

Let B is generated 16 principal components from generator.

Then,

$$error = norm(A_{16} - B) * p_{genrand} * p_{discrand}$$
$$\therefore error = norm(A_{16} - B) * \frac{256}{36058} * \frac{16}{256}$$
$$\therefore error = norm(A_{16} - B) * \frac{16}{36058}$$

Thus the norm generated by subtracting A_{16} and B is then multiplied by the probabilities, resulting into the error between the generated and real image.

VI. FUTURE WORK

The novel approach used here can be extended further by designing a module to generate images back from these given *PCs*. The output *PCs* can be further used to classify the emotions for which the dataset is proposed, this fulfills the task of using GANs to supplement a supervised learning.

REFERENCES

[1] Ian J. Goodfellow,Jean Pouget-Abadie,Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio Generative Adversarial Nets

[2] Alec Radford & Luke Metz, Soumith Chintala Under review as a conference paper at ICLR 2016 Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks

[3] https://medium.com/towards-data-science/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0

[4] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge