# Generative Adverserial Networks for Image Completion

Group 8: Maunil Vyas (1401007), Nirav Akbari (1401008), Neel Puniwala (1401024)
Raj Derasari (1401029), Karan Patel (1401113)
Team EnVyUs
Courses: Algorithms and Optimization for Big Data, Machine Learning
School of Engineering and Applied Science, Ahmedabad University
April 24, 2017

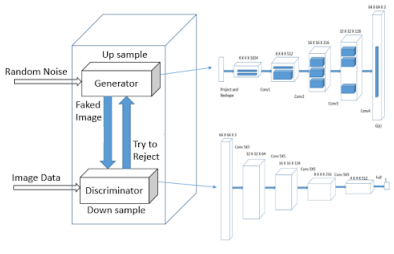
***Abstract*—First introduced by I. Goodfellow in 2014, GAN has become one of the hot topics in the fields of Machine Learning and Artificial Intelligence. In this report, we demonstrate 3 approaches for image generation: Deep Convolutional GAN (DCGAN), and then demo an approach to incorporate PPCA into GAN to make the training efficient. We also try to introduce K-Mean clustering on generalised datasets, to improve the quality of datasets used in training the GAN. Our motive in this project has been to improve on the efficiency of generator while making it less computationally expensive. The results obtained are good, but still, far from desired.**
***Index Terms*—DCGAN, Image Completion, K-Means, PPCA**

## I. An Introduction To GAN

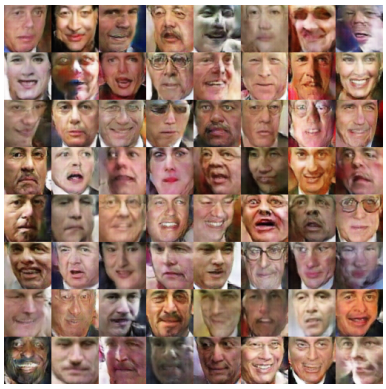
GANs were introduced to machine learning in 2014, and have been embraced. GAN have various applications that currently require a lot of data analysis and computational operations. .
GAN define a system where two neural networks compete to generate data that looks authentic. The generator takes a vector as information and generates images. The discriminator gets tests from both the generator, and the testing dataset, and learns the ability to recognise whether the input is authentic or generated. The generator learns how to create increasingly convincing output whereas the discriminator learns how to improve classification. Simultaneous training of the networks creates the notion that the race between both will push the generated samples to be identical to real data.

## II. Approaches and Results

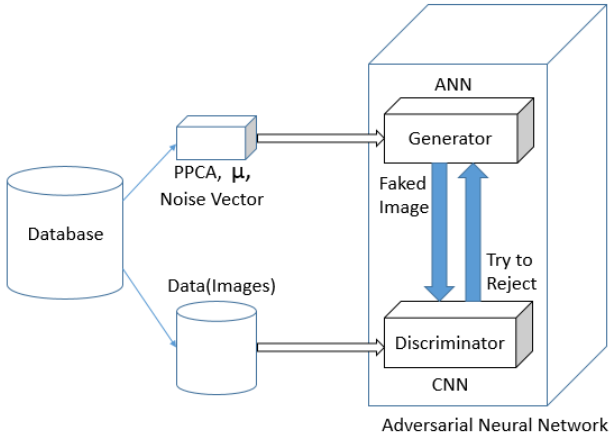
1) DCGAN Implementation for completion of faces: [3]
In this approach, the discriminator and generator are convolutional neural networks (CNN), both 6 layers each. The discriminator has 1 input layer, 4 convolution layers and 1 fully connected output layer. The generator has similar architecture, but in reverse. Given below, is the model in this approach.




Results obtained on the LFW dataset [10] after processing it by [4]:



2) DCGAN Implementation from PPCA of faces
We build from the first approach that was in [3], by reducing the parameters in generating an image. We exploit the property of images, that as a matrix an image can be represented by a linear relation such as $Y = wX + \mu + \epsilon$, where $w$ is the PPCA, $\mu$ is the mean of all images in the dataset and X is the latent variable (produced by the generator). With this model we reduce the complexity of the GAN, and potentially speed up the training time since fewer number of components are required. In our system model, the discriminator is a CNN with 4 hidden layers, and the generator is an artificial neural network (ANN) with 1 hidden layer Given below, is the system model of this approach:

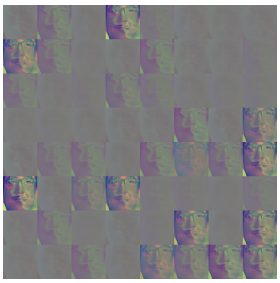


We trained this PPCA-based model with the ML Winter '17 Class Database, and the following results were obtained:

Generated after 50 epochs

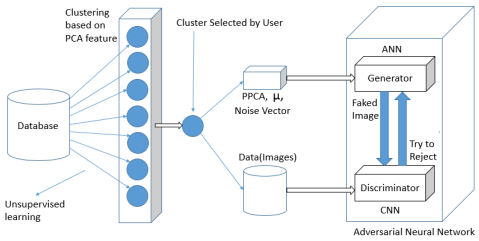
Generated after 230 epochs

We observed that the time taken to generate the image on the right was less than time taken to generate the image in the first approach.
Since this approach was run on an unconstrained/generalized dataset, it is difficult to define faces in the generated image. Hence, we applied this approach a set of faces, bt only of one student. On this small dataset, we obtained the result after 100 epochs:



On this smaller dataset, it can be seen that features of the face from the training data, have been generated by the generator (even, multiple times) so we learnt that it is better to provide data that is similar in features, as a dataset, into the generator. This also makes life tougher for the discriminator as generator ability drastically improves here. Because of limited resources in terms of computation and time, we have not trained or modelled the network enough to generate better colors in images.

3) DCGAN Implementation from PPCA of clustered data
We see that approach (2) fails when the dataset is highly generalized or unconstrained. To improve the quality of the dataset, we apply $K$-means clustering with $K = 4$. We performed this approach on a public face dataset after processing it on Openface [4]. Selecting all 75 images from 1 cluster, we obtained some outputs after 200 epochs. Following, are the system model and results for the same:
System Model:




Observed results:

Generated after 200 epochs

Generated after 2500 epochs

We can see here that the output obtained, has common characteristics to the output mentioned in second approach for training data of 1 student. This meets our expectations, considering that we apply K-Means on the dataset and try to segment larger data into subsets. This would be useful to real world applications that involve separation among inanimate objects in a dataset, such as cars and bikes from a dataset of automobiles, and improving quality of GANs.

## III. Acknowledgements

We would like to acknowledge [2] and [3] for providing us with a starting point for our project. We would also like to acknowledge [4] for providing the Openface repository for face-processing in OpenCV.

## IV. Conclusions

Learning of GAN improves, based on the provided training data. We could observe that when the input is a full image, the time taken for learning is high. Working with PPCA however, reduces the time taken. A straightforward observation is that by increasing the number of epoches, generator loss decreases towards convergence.
Knowing that we have not been able to train the GAN with PPCAs such that the output image is also appropriate in terms of image color – not only the image features – is one of the many possible future improvements.

## References

[1] Goodfellow, 2014: Generative Adverserial Networks
[2] Github | Taehoon Kim on Github
[3] Github | Brandon Amos on Github
[4] Openface repository on Github
[5] Github | Tensorflow Examples
[6] Mnist numbers database
[7] Class Database, Machine Learning, Winter 2017
[8] Public Database for face recognition, Machine Learning, Winter 2017
[9] Introduction to GANs in tensorflow
[10] LFW Faces database
[11] Github — Deshpande, A (2016). Deep Learning Tutorial