Generate Deep Fake Face using DCGAN

Arpit Dhamija
2K18/EC/044
Delhi Technological University
Delhi, India
arpitdhamija_2k18ec044@dtu.ac.in

Dinesh K Vishwakarma
Professor
Delhi Technological University
Delhi, India
dinesh@dtu.ac.in

Gaurav kumar
2k18/SE/060
Delhi Technological University
Delhi, India
gauravkumar_2k18se060@dtu.ac.in

***Abstract*—Deep neural networks are used mainly int the field of supervised learning: classification or regression. Various applications of neural networks are there which have a really great impact in the field of technology. Generative Adversarial Networks or GANs, are one of the most interesting and popular applications of Deep Learning. GANs use neural networks for a very different and unique purpose: Generative modeling. In this paper we worked on DCGAN i.e deep convolutional generative adversarial network and it depicts how convolutional layers can be used with GANs and provides an additional architectural basis for doing Generative modeling. In this study We have worked on a total of three datasets for the implementation of dcgan and then after training of our model we have successfully been able to generate deep fake faces.**

Keywords—DCGAN, Generator, Discriminator, Relu, Random Input Vector

I. Introduction

Generative adversarial networks (GANs) are the latest innovation in computer vision and machine learning. Generative adversarial networks are *generative* models: they generate new instances of data that resemble our training data. For example, GANs are capable of creating images that look exactly like photographs of human faces, even though those faces don't belong to any real person.Generative modeling comes under the field of unsupervised learning task in machine learning that comprises automatically discovering and learning the regularities or patterns in input data in such a unique way that the generated model can be used for the process of generation of images or output new examples that plausibly could have been fetched from the original dataset taken for this process..
In this project, we will be generating Deep Fake Human Faces, Anime faces and Handwritten digits which will look realistic. We have implemented these all using DCGAN(Deep convolutional Generative Adversarial Networks). Initially we generate a random noise using a generator and pass that random noise into our model and that model is improved by continuous training of GANs. And after a certain amount of epochs, we get very realistic images that are basically deep fake images.

II. Related work

GANs have a wide research history and a vast array of practical applications.There is so much research has been conducted so far for the GAN which uses the different models to generate fake faces and images using dcgan and other types of gans. Below are some findings of GAN in table 1 which are showing different methods of image generation using different models on different datasets.We have included past work of a total of 5 papers in this paper. Based on our previous research papers, dcgan has come out as the best model to generate faces.Also in Mirza and Osindero's paper they claimed CGAN to be the theme to start of art of GAN.
In our finding many datasets such as CIFAR10, MNIST , Imagenet-1, CelebaA are used. Karras et al. have worked for the improvement of features such as quality, stability and variation of GAN models using progresive growth. Past papers contain various features and technologies such as feature matching, minibatch discrimination, Image Labelling, minibatch standard deviation,SWD, MSProp etc.
All of this work has been summarized in the following table.

Author(s)	Proposed Model	Dataset	Features
Radford et al. (2015)	Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks	Imagenet-1k	Visualizing GAN features, Latent space interpolation, using discriminator features to train classifiers,LSUN
Salimans et al. (2016)	Improved Techniques for training GANs contains the best hypotheses for GAN instability.This paper gives many additional tech features designed for the process of stabilizing the training of DCGANs.	MNIST, CIFAR-10 and SVHN	feature matching, minibatch discrimination, historical averaging, one-sided label smoothing, and virtual batch normalization
Mirza and Osindero (2014)	Conditional Generative Adversarial Networks(CGAN). Introduction of a conditional version of GAN.It shows how integrating the class labels of data results in more stable GAN training.	MNIST	Image Labelling, Unimodal, Multimodal, Focusing on image-to-image and text-to- image
Karras et al. (2017)	Progressive growing of GANs for Improved Quality, Stability and VariationThis paper uses a multi-scale architecture where the GAN builds up from 4^2 to 8^2 and up to 1024^2 resolution.	CelebaA,CIFAR10 ,LSUN Bedroom	target image resolution ,SWD,MSSSIM,RMSProp,Inception Score, Minibatch standard deviation,Normalization in generator and discriminator

Liu, Shouqiang et al. (2019)	The research of virtual face based on Deep Convolutional Generative Adversarial Networks using TensorFlow	CelebaA	DCG-CNN,SVM,VGG16,Alex-Net,ZFNet

Table 1: Previous Researches on GAN

III. Methodology

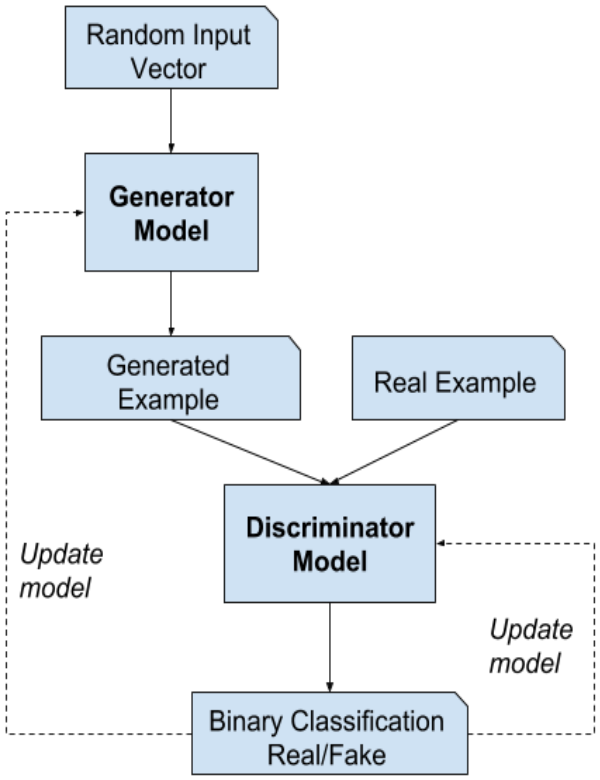


Fig.1 Experimental Design

A. Data Collection

For the Experiment of generation of deep fake faces we have implemented our model on the following three datasets :-

1. **MNIST Handwritten digits**
   We have directly loaded this dataset from Keras library.This dataset consists of 60k 28x28 grayscale images of the 10 digits, along with the test set of 10,000 images. It comprises images of handwritten digits from 0 to 9.
2. **Anime Faces dataset**
   We have collected this data from kaggle. This dataset contains a total of 63.2k anime faces. This dataset is often used for image generation or image detection projects with anime faces.
3. **CelebA dataset** (img_align_celeba)
   We took this dataset from kaggle. It is one of the directories of the celebA dataset set named img_align_celeba.
   It contains a total of 203k jpg files containing images of celebrity faces.
   Information about this img_align_celeba directory is given below:

.jpg files : 203k
.csv files: 4
columns : 59(Integer:55, String:4)

B. Data Processing and visualization

We've loaded data using the ImageFolder class from torchvision. We will also crop and resize the images to 64x64 px, and normalize the pixel values with a mean & standard deviation of 0.5 for each channel. This will ensure that pixel values are in the range [-1 to 1], which is more convenient for training the discriminator. We will also generate a data loader to load the data in batches.

This is a batch of 64 images taken out of main data which we'll for showing our results and this is fed into the network after converting it into the random input vector.

C. Generative Adversarial Network

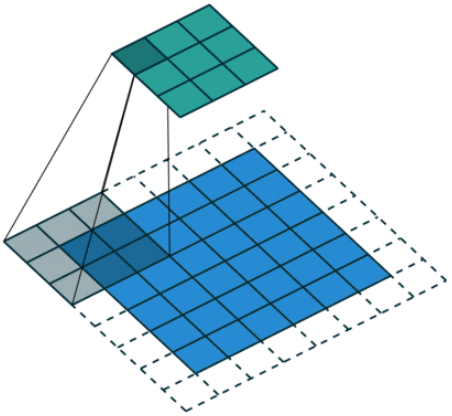
There are two neural networks: a *Generator* and a *Discriminator*. The task of the generator is to generate a "fake" sample given a random vector/matrix, and the discriminator attempts or endeavors to detect whether a given sample is "real" (picked from the training data) or "fake" (generated by the generator). Training happens in tandem or barouche: we trained the discriminator for a few epochs, then trained the generator for a few epochs, and repeated. So by doing this process again and again, both the generator and the discriminator got improvised at doing their work.

GANs however, can be notoriously or extremely difficult to train, and they are extremely sensitive to hyperparameters, activation functions and regularization.

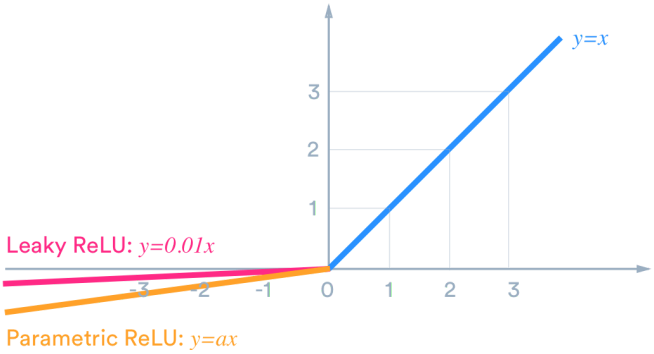
Deep Convolutional Generative Adversarial Network

Discriminator Network

The discriminator takes an image as input, and tries to identify it as "real" or "generated". In this sense, it's like any other neural network. We'll use a convolutional neural network (CNN) which outputs a single number output for every image. We'll use stride of 2 to progressively reduce the size of the output feature map.



Note that we are using the Leaky ReLU activation for the discriminator.

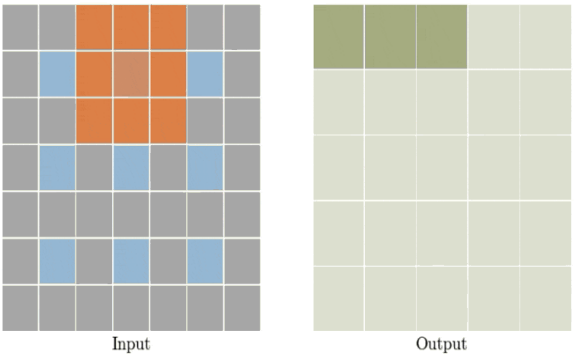


Different from the regular ReLU function, Leaky ReLU allows the pass of a small gradient signal for negative values. As a result, it makes the gradients from the discriminator flow stronger into the generator. Instead of passing a gradient (slope) of 0 in the back-prop pass, it passes a small negative gradient.

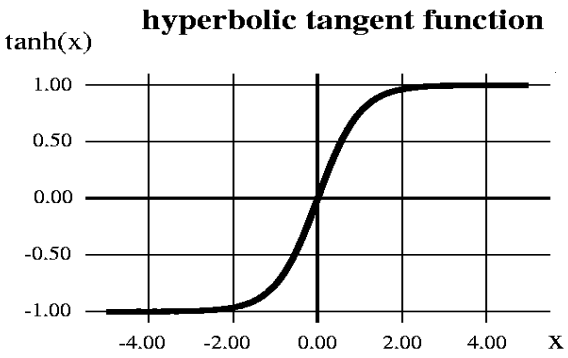
Just like any other binary classification model, the output of the discriminator is a single number between 0 and 1, which can be interpreted as the probability of the input image being real i.e. picked from the original dataset.

Generator Network

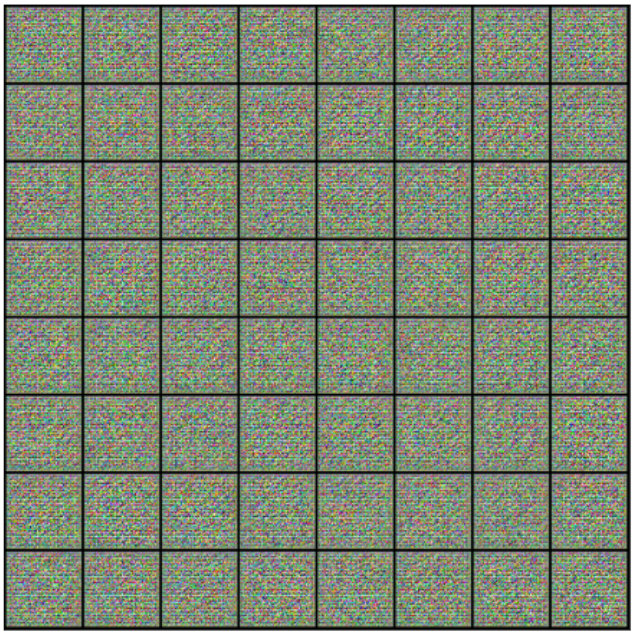
The input to the generator is typically a vector or a matrix of random numbers (referred to as a latent tensor) which is used as a seed for generating an image. The generator will convert a latent tensor of shape (128, 1, 1) into an image tensor of shape 3 x 28 x 28. To achieve this, we'll use the ConvTranspose2d layer from PyTorch, which is performed as a transposed convolution which is also known as **deconvolution**.



We use the TanH activation function for the output layer of the generator.

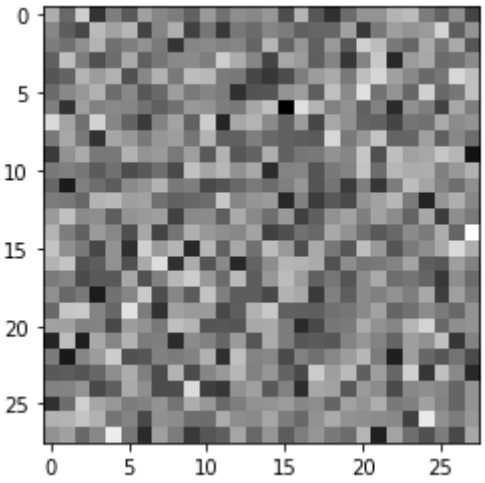


Note that since the outputs of the TanH activation lie in the range [-1 to 1], we have applied the similar transformation to the images in the training dataset. Let us generate some outputs using the generator and view them as images by transforming and denormalizing the output.



This is our initial batch of random noise

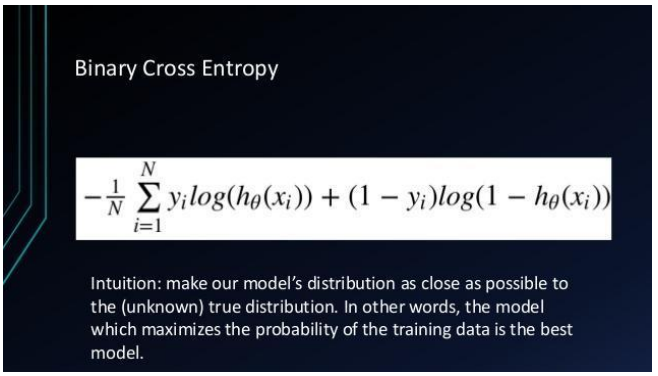
As one might expect, the output from the generator is basically random noise, since we haven't trained it yet.



This is our image of random noise of MNIST dataset. As one might expect, the output from the generator is basically random noise.

Discriminator Training

Since the discriminator is a binary classification model, we can use the binary cross entropy loss function to quantify how well it is able to differentiate between real and generated images.



Here are the steps involved in training the discriminator.

1. We expect the discriminator to output 1 if the image was picked from the real MNIST dataset, and 0 if it was generated using the generator network.
2. We first pass a batch of real images, and compute the loss, setting the target labels to 1.
3. Then we pass a batch of fake images (generated using the generator) , pass them into the discriminator, and compute the loss, setting the target labels to 0.
4. Finally we add the two losses and use the overall loss to perform gradient descent to adjust the weights of the discriminator.

Note that we don't change the weights of the generator model while training the discriminator.

Generator Training

Since the outputs of the generator are images, it's not obvious how we can train the generator. This is where we employ a rather elegant trick, which is to use the discriminator as a part of the loss function. Here's how it works:

1. We generate a batch of images using the generator, pass the into the discriminator.
2. We calculate the loss by setting the target labels to 1 i.e. real. We do this because the generator's objective is to "fool" the discriminator.
3. We use the loss to perform gradient descent i.e. change the weights of the generator, so it gets better at generating real-like images to "fool" the discriminator.

Here's what this looks like in code.

Now, create a directory where we can save intermediate outputs from the generator to visually inspect the progress of the model. We will also make a helper function to export the generated images.

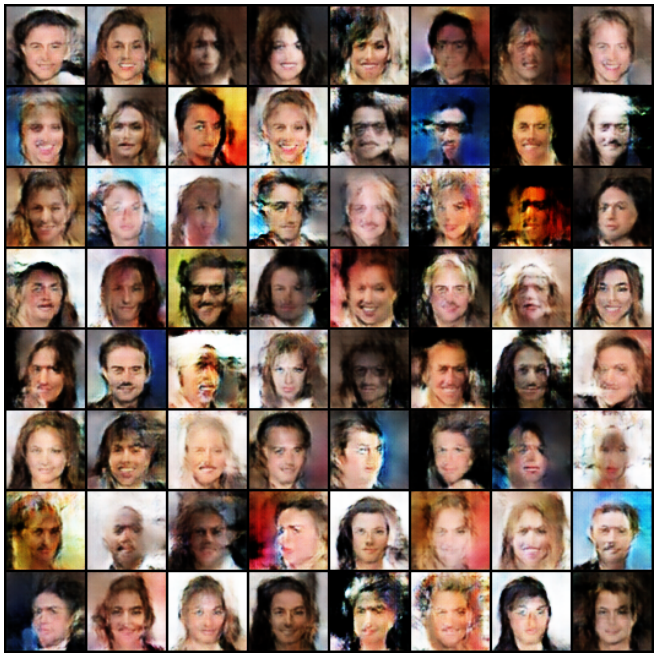
We'll use a fixed set of input vectors to the generator to see how the individual generated images evolve over time as we train the model.

Now, let's save one set of images before we start training our model.

Full Training Loop

Let's define a fit function to train the discriminator and generator in tandem for each batch of training data. We will use the Adam optimizer with some custom parameters (betas) that are known to work well for GANs. We will also save some sample generated images at regular intervals for inspection.

IV. RESULTS AND ANALYSIS

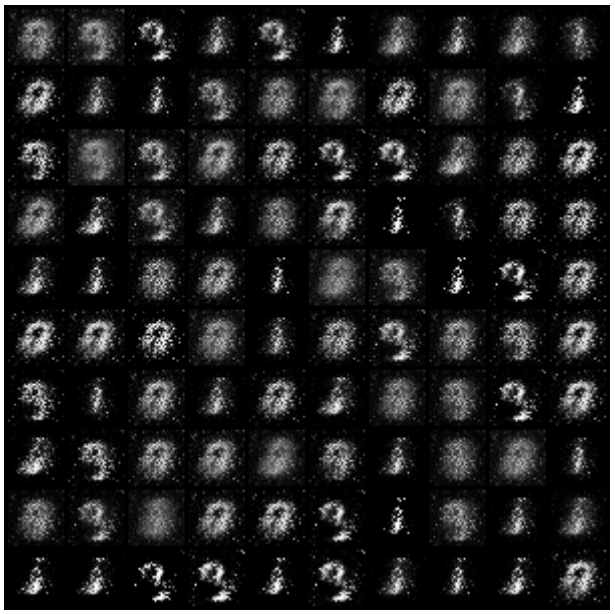


This is the output of the 1st **epoch** of training. These are the generated Fake human Faces. This will improve after every epoch.

This is the output of the 25th **epoch**. This is a very improved version. We can get even more realistic images if we do more epochs training.

This is the output of the 1st **epoch** of training. These are the generated Fake human Faces. This will improve after every epoch.

This is the output of the 25th **epoch**. This is a very improved version. We can get even more realistic images if we do more epochs training.



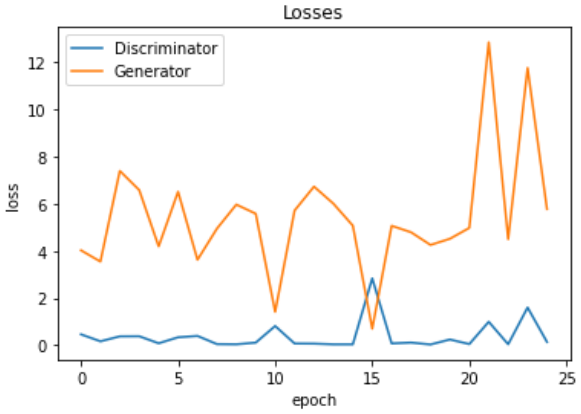
This is the output of **10th epoch** of MNIST dataset

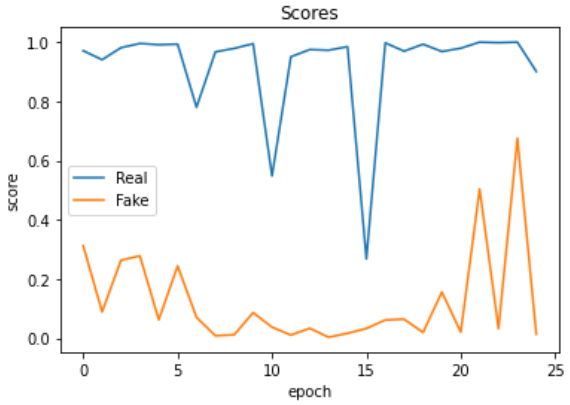
This is the output of the 300th **epoch**. This is a very improved version. We can get even more realistic images if we do more epochs training.
We can visualize the training process by combining the sample images generated after every epoch into a video using OpenCV.

We can also visualize how the loss changes over time. Visualizing losses is quite useful for debugging the training process. For GANs, we expect the generator's loss to reduce over time, without the discriminator's loss getting too high.

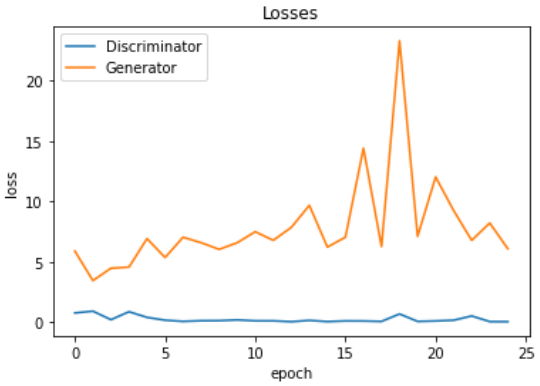
Graphs of Losses and Scores

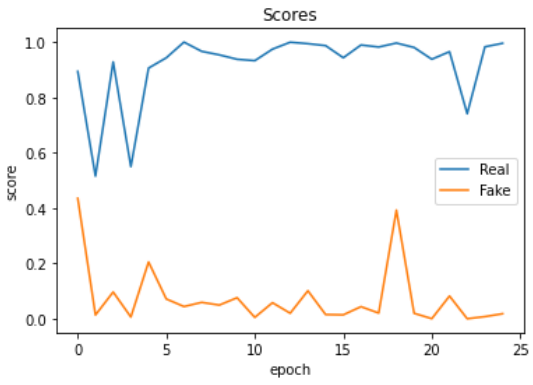
1. Real Human Faces



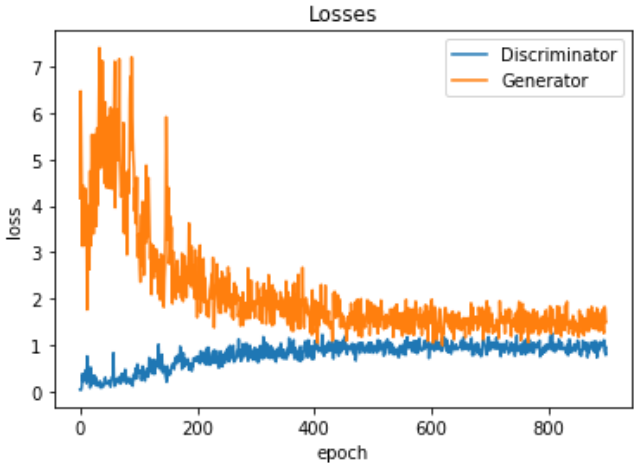


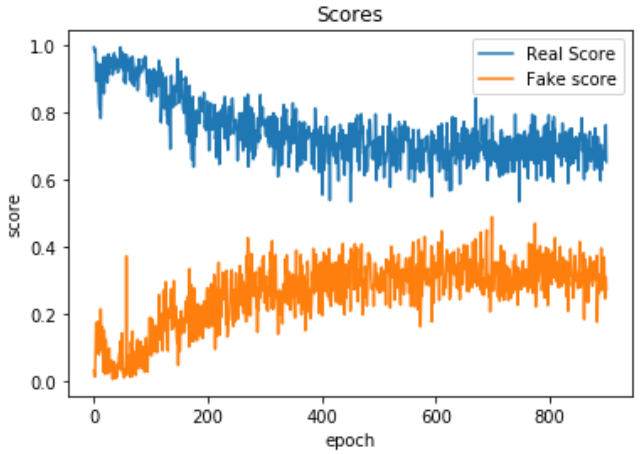
2. Anime faces





3. MNIST





V. CONCLUSION

After the existence of gans it's become really very easy to generate images. It has added a new dimension to the technology. Throughout the journey of our work we have Learned the architecture of deep convolutional generative adversarial networks. Our model makes use of various features of the image to generate fake images using the features of the image given the model. DCGAN can be able to generate the image to that much accuracy which we can't figure out whether this image is real or not. DCGAN can be beneficial on large datasets as well. Experimental results show that our model has better accuracy. In addition to generation of images the discriminator will be quite successful in detecting the image whether an image is fake or not.

We know that gans have tremendous in today's world but along with a lot of applications there are few disadvantages which are associated with gans. It's become very easy doing image masking using these kinds of technologies and it's becoming a good tools for the criminals and terrorists.

So after all we should use technology for good purposes and technologies like dcgan are no different than magic in real work.

VI. FUTURE WORK

In this project we have implemented the model of dcgan.

In future we'll try to develop a better architecture of dcgan than this one along with this we will also work on a few more GANs such as StyleGAN, CGAN, InfoGAN. We will do a comparative study on these gans and we'll try to implement this comparative study on more than 2 dataset so that we can get more accurate comparison as sometimes an algorithm overfits only for a particular dataset so performs best in that case not else.

We are also thinking of working on smoke detection techniques using dcgan and we'll use an advanced technology for this such as FlowNet.

At last in future we will try to work on the areas which can help in better life such as working in the area of smoke detection.

VII. REFERENCES

I. Bergstra, James and Bengio, Yoshua. Random search for hyper-parameter optimization. JMLR, 2012. Coates, Adam and Ng, Andrew. Selecting receptive fields in deep networks. NIPS, 2011.

II. Coates, Adam and Ng, Andrew Y. Learning feature representations with k-means. In Neural Networks: Tricks of the Trade, pp. 561–580. Springer, 2012.

III. Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009.

IV. IEEE Conference on, pp. 248–255. IEEE, 2009. Denton, Emily, Chintala, Soumith, Szlam, Arthur, and Fergus, Rob. Deep generative image models using a laplacian pyramid of adversarial networks.

V. arXiv preprint arXiv:1506.05751, 2015. Dosovitskiy, Alexey, Springenberg, Jost Tobias, and Brox, Thomas. Learning to generate chairs with convolutional neural networks. arXiv preprint arXiv:1411.5928, 2014.

VI. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. Generative adversarial nets. In NIPS, 2014.

VII. Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. arXiv preprint arXiv:1506.05751, 2015.

VIII. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

IX. Ian J Goodfellow. On distinguishability criteria for estimating generative models. arXiv preprint arXiv:1412.6515, 2014.

X. Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. arXiv preprint arXiv:1602.05110, 2016.

XI. Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. arXiv preprint arXiv:1603.07442, 2016.

XII.