GANs: Handwriting Profiling

Swapnil Bembde, Ritvik Mathur

Electrical Engineering, IIT Bombay
swapnilbembde@gmail.com
ritvikmathur1501@gmail.com

Abstract— **This project aims at expanding any current dataset by generating similar images using GANs concept. This expanded dataset will contribute to other machine learning problems. We take random noise as an input and transform into usable handwriting texts. This generation is a result of 2 player game. we train our 2 players of the game-generator and discriminator with the real images. The output obtained are the sets of images which are computer generated handwritten numerals.**

Keywords— **Generative Adversarial Networks, Binary Cross Entropy, Neural Networks, Supervised Learning, MNIST, EMNIST.**

I. INTRODUCTION

Handwriting is an important skill developed and mastered by the human beings over the course of time. It is an essential tool used to communicate and interpret the actions of certain individuals. However, even though the process of learning to write is similar across different parts of the world, still most of the people have considerably different writing styles and visibly different handwritings. So, it is not easy even for other humans to mimic a person's handwriting.

In the pretext of Machine Learning, it is important that we obtain a good accuracy for any model. For this, there should be sufficient amount of properly classified data, since a good quality data is required for better learning in any models. However, in many cases, we are not able to obtain a very large dataset. Even in cases where there are a lot of classes, the dataset gets divided further, leading to even lesser data corresponding to each class. This makes training such models very difficult, and output unreliable for projection in the real world.

If we can expand the dataset corresponding to the given datasets, so that the properties comply with the original, and no misclassified or noisy data is generated, then we can extend the datasets for any system to obtain an overall larger dataset, which can be used as a good training dataset to get a good and accurate learning model.

We focus on handwriting-based learning problems. So, the main aim of this project is to generate computer-based handwriting samples, using initial smaller number of samples. For implementing this, we are using the concept of GANs to take an initial random noisy image, and apply transformations to it using NNs, to output a generative image. The NN is updated iteratively after the generated image is distinguished with respect to the real samples.

II. BACKGROUND

The Generative Adversarial Nets was proposed in [2] by Ian J. Goodfellow and his fellow researchers in 2014. It was an innovative idea involving the concept of the two-player game as a learning algorithm. Until this time, all the generative models focussed on some parametric specification, which were then used to maximize the log likelihood, and the models were trained. However, these models require numerous approximations for being able to solve. In GANs, the derivatives are backpropogated through the generative networks.

Handwriting recognition is a very frequent and in-demand application of machine learning. Currently, the most common method is to use Recurrent Neural Networks (Graves et al. 2008). Handwriting generation is a new concept to apply via extension of the datasets.

III. SYSTEM MODEL

We have used the basic and origin of all the architectures which is developed by Goodfellow et. al. [2]. This interesting idea is based on the concept of generative machine eliminating the must use method, Markov chains used in stochastic networks. In GANs we don't work with an explicit density function, but instead game-theoretical approach is used. Most straightforward method to implement adversarial networks are multilayer perceptrons. Hence both generator and discriminator are made of a neural network with some activation functions at each layer. The purpose of generator is to take in a noise vector and generate an image similar to real images. This generation has a motivation of fooling the discriminator by doing the same. Whereas, discriminator classifies true images and fake images.

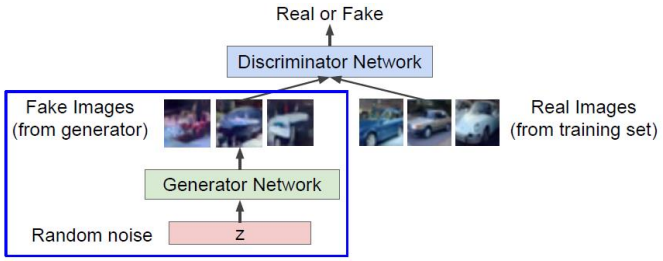

Fig. 1 Flow Graph for a GAN

A. Generator

Generator network takes random noise as an input and this will generate a sample from the training distribution. So, this every input of random noise will correspond to a sample from

the training distribution. Using random noise, generator is going to fool the discriminator by generating real looking binary images, i.e., each of the pixels is of binary value.

The Generator NN is a three-layered Network for MNIST and 4 layered for EMNIST. We have different model for EMNIST because the dataset is far bigger in size as compared to MNIST. We generate random images and give batch-wise input to the generator. Each layer has ReLu as an activation function except last layer, last layer has hyperbolic tan function. Input layer has 100 neurons. The output is 784*1 sized vector. The output image consists of an image with 16 sub-images in case of MNIST, each having 1 generated character. Loss function used is Binary Cross Entropy. We try to minimize the Loss function corresponding to it. This is done by updating the parameters of the NN after each iteration.

B. Discriminator

The Discriminator takes 2 sets of images as the input - Real Image Samples (from the Training set) and Computer-Generated Samples from the Generator. The task of the Discriminator is to try being able to successfully distinguish between these two types of data. The objective of the Discriminator is to minimize the loss function corresponding to this. Its aim is to not be fooled by the Generator.

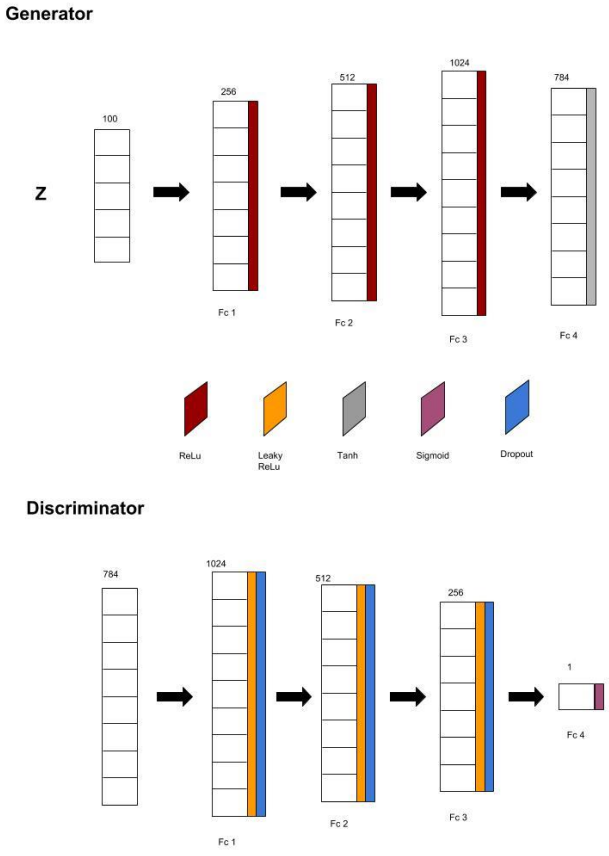

Fig. 2 Models for Generator and Discriminator

Even the Discriminator is a three-layered Neural Network for MNIST and four-layered for EMNIST. The model is quite similar to generator's. However, the input here is the output of the generator, as well as the samples from the real images, given as input randomly. Also, we are using Leaky ReLu as activation function for each layer except last layer and the last layer has Sigmoid activation function. Also, we have included dropouts after each layer in this. The discriminator minimizes its loss function, which depends on the classification of Real and Generated images (Binary Cross Entropy). This leads to improvement in the Discriminator after each Epoch.

C. Interpretation

The idea behind this method is to work like a 2-player game. For example, if we take the game of chess. In each turn, 1st player tries to minimize the chances of the second player winning, by trying to disqualify the pieces with the pieces they themselves have. In the consecutive turn, the other player tries the same, but with the remaining set of pieces with respect to both the parties. They need to constantly update their strategies as the game progresses.

This is similar to the concept of GANs. In that, alternatively the Generator and Discriminator try to maximize their objective. First, the discriminator minimizes the loss function with respect to the image sets, then the generator updates it's NN to generate images which try to maximize the current loss function of the discriminator, while minimizing its own loss function. Repetitive steps lead to the improvement in the generated images, enough so that the best distinguisher is not able to distinguish between them.

D. Algorithm

The goal of our system is as follows

$$min_{\theta_g} max_{\theta_d} [\mathbb{E}_{p\sim data} log D_{\theta_d} + \mathbb{E}_{z\sim p(z)} log(1 - D_{\theta_d}\left(G_{\theta_d}(z)\right))]$$

To achieve this, we break our algorithm into the following parts, for Discriminator and Generator respectively

Gradient descent on Discriminator

$$max_{\theta_d} [\mathbb{E}_{p\sim data} log D_{\theta_d} + \mathbb{E}_{z\sim p(z)} log(1 - D_{\theta_d}\left(G_{\theta_d}(z)\right))]$$

Gradient ascent on Generator

$$min_{\theta_g} [\mathbb{E}_{z\sim p(z)} \log\left(D_{\theta_d}\left(G_{\theta_d}(z)\right)\right)]$$

We keep repeating this till we observe significant convergence in the loss functions. The end result we expect is to get Generated images very similar to actually real images. In our case, this refers to Generative images being indistinguishable from actual handwritten text.

Every iteration, first the Discriminator loss gets updated, which updates the NN, then the Generator loss gets updated, which depends on Discriminator results for all the images classified and updates its NN. Every image is checked in every iteration, batch-wise. The loss functions of both the Discriminator and the Generator converge after some iterations. This way, we are able to have a Generator, which can generate image even a very good Distinguisher can't distinguish with high probability.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(\boldsymbol{x}^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(\boldsymbol{z}^{(i)}))) \right]$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(\boldsymbol{z}^{(i)})))$$

end for

Fig. 4 Pseudocode for a GAN algorithm

IV. Datasets for Discriminator

Handwriting. For testing the validity of our system over various kinds of datasets, we have tested our program over different datasets, pulled from different publicly available sources. Following datasets were used

A. *MNIST*

This is a common image formatted dataset, consisting of 60,000 handwritten digits as training set, and 10,000 handwritten digits as testing set. We are interested only in the training set, since we don't have any application of testing in our method. The collection consisted of taking handwritten samples from 500 different writers. This was used as an initial dataset to check the validity of our program, to later include the alphabetic characters as well. Refer to [4] for dataset formatting and further details.

B. *Chars 74K*

This dataset consisted of 7705 handwritten samples. This dataset had characters consisting of digits and alphabets, both small and capital. The collection consisted of images with clear alphabets obtained through various placed cameras, on places like board or banners. Refer to [5] for further details on the dataset generation and processing. [6] is the link where datasets can be downloaded.

C. *EMNIST*

This dataset is the extension of MNIST Dataset. In addition to the handwritten digits as the character, it also consists of alphabets, both small and large. Refer to [7] for dataset formatting and further details.

V. Experiments

We have taken different datasets as described in section III and applied the concept of GANs on them. We set our program for 100 back and forth iterations. The following experiments were performed.

A. *MNIST Dataset*

The preliminary tests consisted of only numeric characters from the MNIST datasets. The images generated were visibly similar to handwritten digits. The results are shown below

We can visibly view that the characters obtained are visibly recognizable as handwritten digits.

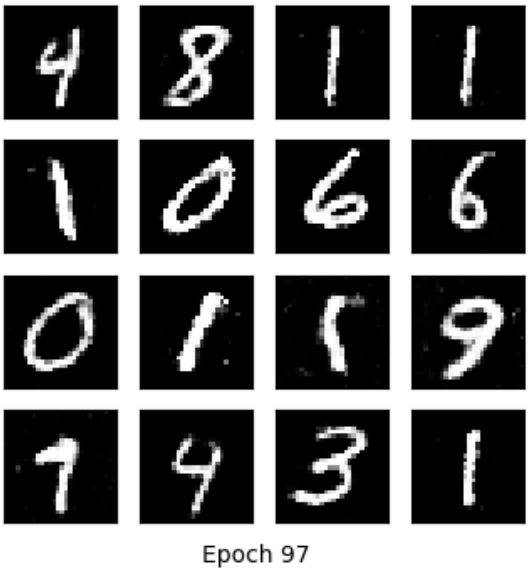

Fig. 5 Image generated for the MNIST in the last epoch

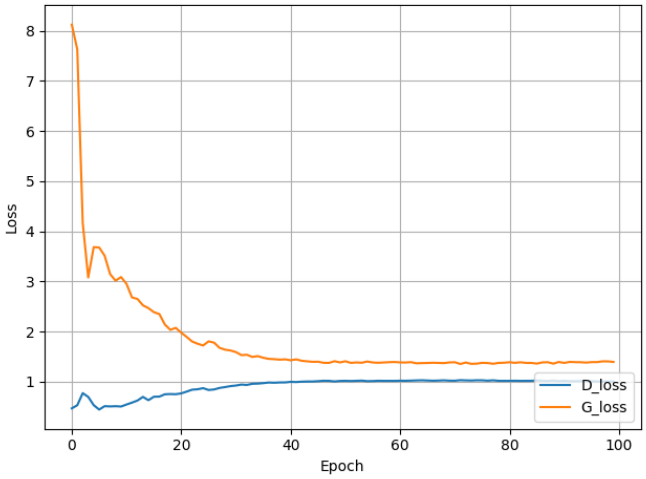

Fig. 6 Training Histogram for Generator and Discriminator

We can see that the rates converge till about 80 epochs.

B. *Chars 74k Dataset*

This data consisted of alphabets, but the samples for each character were very low. We only had 55 samples for each class, while we have 26 classes (for each alphabet a-z). This data turned out to be very less for proper training, so the generated images were very noisy. So, Chars 74k Dataset was not successfully extended using GANs.

C. *EMNIST Dataset*

This data, an extension of the MNIST, contained handwritten alphabets. We have taken the mixed bi… dataset, which merges some similar alphabets, considering a class of

only 47 characters, including 0-9, a-z and A-Z. The similar classes are removed from these. The results obtained for these were as follows.

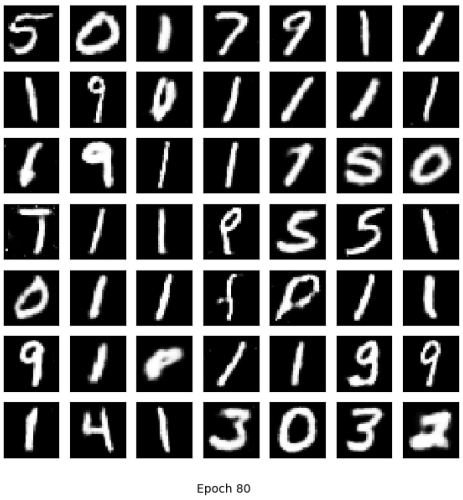

Fig. 7 Image generated for the EMNIST in the last epoch

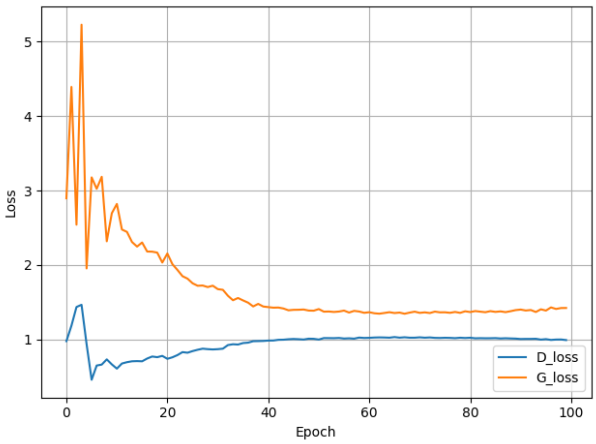

Fig. 8 Training Histogram for Generator and Discriminator

We can clearly see the characters closely representing handwritten characters. Many of the characters are numeric, due to dataset being biased more towards the numeric characters. However, we can observe a many alphabetic characters too.

The Loss rates for Generator and Discriminator converges till about 70 epochs. Initially there are a lot of oscillations in the both the losses, because there is a lot of data to process. However, after some iterations, the rates stabilize.

VI. Conclusions and Future Work

We can easily visually recognise most of the different handwritten characters in our generated character images. The results[1] obtained displayed the decent ability of the GANs to learn human-handwriting characteristics. This can be used to extend the datasets for an MLP significantly, provided that sufficient data is provided.

We hope to improve the performance further by using DCGAN rather than vanilla GAN. Increasing the number of layers of the model can further improve the results. We can expect much more deeper linkages to the real images, to obtain a more leaner generation.

References

[1] Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR* abs/1511.06434

[2] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets. *NIPS*, 2014.

[3] Isola, Phillip & Zhu, Jun-Yan & Zhou, Tinghui & Efros, Alexei. (2017). Image-to-Image Translation with Conditional Adversarial Networks. 5967-5976. 10.1109/CVPR.2017.632.

[4] http://yann.lecun.com/exdb/mnist/

[5] T. E. de Campos, B. R. Babu and M. Varma. Character recognition in natural images. In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP), Lisbon, Portugal, February 2009

[6] http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

[7] https://www.nist.gov/itl/iad/image-group/emnist-dataset

1 Github Repository Link for Code – https://github.com/swapnilbembde/Handwriting_profiling/tree/master/emnist