TEST Engineering & Management

September/ October 2020
ISSN: 0193-4120 Page No. 9- 13

HANDWRITTEN WORD GENERATION USING GENERATIVE ADVERSARIAL NETWORKS

Shilpa Shinde[1], Sujay Rokade[2], Ruchita Nagare[3], Amit Panthi[4]

[1,2,3,4] Department of Computer Engineering, Ramrao Adik Institute of Technology, DY Patil Deemed to be University, Navi Mumbai, Nerul, Navi Mumbai, India.

Article Info
Volume 83
Page Number: 9-13
Publication Issue:
September/October 2020

Article History
***Article Received**: 4 June 2020*
***Revised**: 18 July 2020*
***Accepted**: 20 August 2020*
***Publication**: 15 September 2020*

Abstract

We propose a framework for generating handwritten samples using Generative Adversarial Networks. This is a slightly different version of the Generative Adversarial Networks in which two networks are pitted against each other in an adversarial format to maximize the "accuracy" of the generated content. Instead of that, we introduce a third network that controls the output of the generator and enforces legibility. We train this architecture on the famous IAM dataset, in order to replicate handwritten words. The generated words bear some resemblance to the words in the dataset. Although not perfect, we plan to shed some light on the topic using this architecture and further improve the model using advanced training methods and new techniques to help the model converge faster.

1. Introduction

Using Unsupervised Learning for learning data distributions is quite powerful because you don't have to provide it with an explicitly defined problem statement, as it figures out all the nuances of the given data by itself. Quite some research has gone into this field, as newer architectures such as Variational Autoencoders have come up and have been used in applications such as Image Filling. Although, the idea of Generative Adversarial Networks has been quite revolutionary in particular. Yann LeCun called GANs, "the most interesting idea in the last 10 years of machine learning". GANs have seen work in varied fields, from medical to image processing. GANs have also been used in the past to implement style transfer, an image manipulation style wherein the 'style' of the first image is used to recreate the second image. We try to use a GAN model to generate the handwritten image of an input word.

2 Literature Survey

This paper deals with the identification of sequences in unstructured data. One of the first major breakthroughs in this field came with the introduction of Long Short Term Memory cells [12] (LSTM, a type of GAN[2]), proposed by Sepp Hochreiter and Jurgen Schmidhuber in 1997. The cells could remember very long sequences, in O(1) complexity local in both time and space. However, LSTM could not deal with temporal data, and it was computationally expensive.

The paper "Sequence Transduction with Recurrent Neural Networks" [13] (Graves, 2012) dealt with the issue of handling temporal data by transforming input sequences into output sequences invariant to sequential distortions such as shrinking, stretching, etc. However, this uses two separate networks to process the input and output sequences, when it is usually more desirable to have a single network with all the information.

Graves then produced another model in 2013, which used LSTM cells and Mixed Density Networks in conjunction in his paper "Generating Sequences with Recurrent Neural Networks" [14] which showed significant results in the field of online generation; the only drawback being its

compatibility only with online data.
“Synthesizing and Imitating Handwriting Using Deep Recurrent Neural Networks and Mixture Density Networks” [15] (2018), published by K. Manoj Kumar, Harish Kandala and Dr. N. Sudhakar Reddy shows the successful implementation of Graves’ model. The model outputted a variety of handwritings based on several tunable hyperparameters.
In 2019, yet another model was proposed by Eloi Alonso, Bastien Moysset, and Ronaldo Messina in their paper “Adversarial Generation of Handwritten Text Images Conditioned on Sequences” [1] which used Generative Adversarial Network, Bidirectional LSTM, and Self Attention GAN Layers together. This model was able to generate fairly recognizable word images after being trained on the RIMES and OpenHaRT datasets and could be trained on offline handwriting data. However, the data preprocessing required for this model was cumbersome and prone to human error.
As stated previously, we discovered that there are two types of models we can create. First is the purely sequential model, which consists only of RNN architecture. It takes as input online handwriting data, which is just a sequence of points on the 2D plane. We found this approach to be impractical because this is not the most abundant form of data available. The most abundant form it is available in is the offline format, i.e. scanned images of handwritten samples. And we have decided to implement a model, the input to which will be offline handwritten samples. For this, we use a variant of Generative Adversarial Networks that is discussed in the following sections of this report.

3 Methodology

We create a generative neural network that is capable of learning the task of converting a textual word to a handwritten image (based on Alonso et al’s[1] architecture). We do this by creating a very delicate and volatile setting consisting of four neural networks that aid the generative network to learn the task and help it converge.
The system consists of four principal components:

1. Language Model
2. Generator
3. Discriminator
4. Recognizer

The Language Model is simply a Bidirectional LSTM model with 4 stacked layers

3.1 Generator

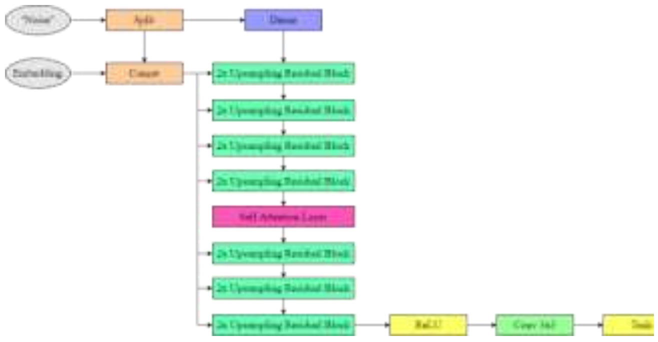
"Noise"
Split
Dense
Embedding
Concat
2x Upsampling Residual Block
2x Upsampling Residual Block
2x Upsampling Residual Block
2x Upsampling Residual Block
Self Attention Layer
2x Upsampling Residual Block
2x Upsampling Residual Block
2x Upsampling Residual Block
ReLU
Conv 3x3
Tanh

Fig. 1. Generator

The Generator consists of a 2x Upsampling Residual Block that constitutes a major part of the neural network. Input to the generator is a” noise” vector, concatenated with the embedding received from the language model. The noise is split into 8 chunks concatenated with the embedding and passed to every upsampling block. The first split is passed to a Dense layer and then forwarded to the up-sampling residual block. We also incorporate a Self-Attention layer between the 4th and the 5th block. The self-attention layer serves to provide global coherence to the generated images. We then finish off the network with a ReLU activation, a final 3x3 Convolutional layer, and a tanh activation layer.

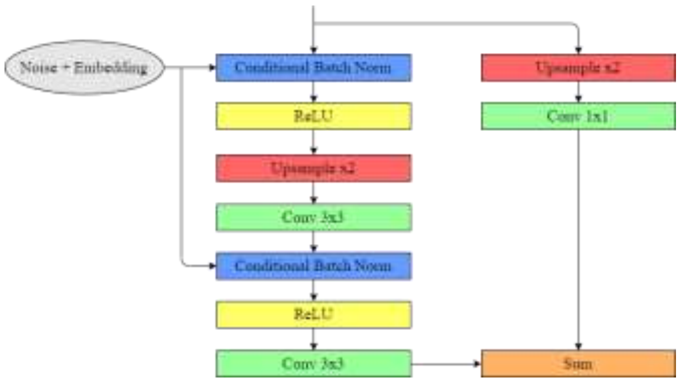

Fig. 2. 2x Upsampling Block

The Generator consists of a 2x Upsampling Residual Block that constitutes a major part of the neural network. Input to the generator is a” noise”

vector, concatenated with the embedding received from the language model. The noise is split into 8 chunks concatenated with the embedding and passed to every upsampling block. The first split is passed to a Dense layer and then forwarded to the upsampling residual block. We also incorporate a Self-Attention layer between the 4th and the 5th block. The self-attention layer serves to provide global coherence to the generated images. We then finish off the network with a ReLU activation, a final 3x3 Convolutional layer, and a tanh activation layer.

3.2 Discriminator

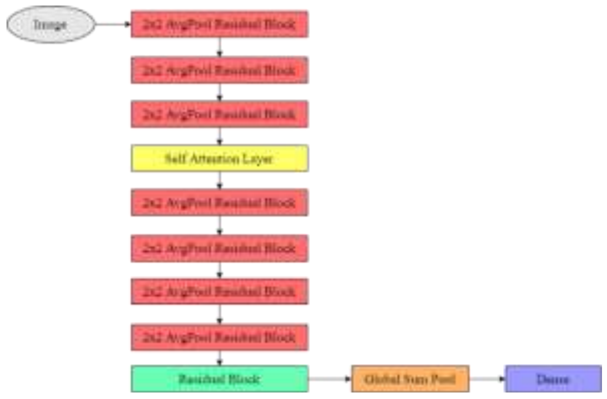
Image
2x2 AvgPool Residual Block
2x2 AvgPool Residual Block
2x2 AvgPool Residual Block
Self Attention Layer
2x2 AvgPool Residual Block
2x2 AvgPool Residual Block
2x2 AvgPool Residual Block
2x2 AvgPool Residual Block
Residual Block
Global Sum Pool
Dense

Fig. 3. Discriminator

The Discriminator consists of a block similar to the 2x Upsampling Residual Block, which is the 2x2 Average Pool residual block. The function of this block is the exact same as the Generator counterpart, but it downscales the input by a factor of 2. The self-attention layer in this network also serves the same purpose but is placed between the 3rd and the 4th 2x2AvgPool Residual Block. We then finish off this network with a Global Sum Pool layer and a final Dense layer. The architecture of the Average Pool Residual Block is discussed below.

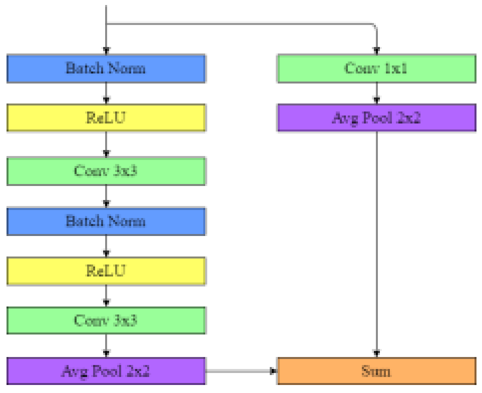

Fig. 4. 2x2 Average Pooling Residual Block

The block starts off with a batch normalization layer, into a ReLU activation. The output is then passed to a 3x3 Convolutional layer and the process repeats. The 2x2 Average Pooling layer is placed at the end of the chain to downscale the output by a factor of 2. There is a residual connection just like the upsampling counterpart.

The Recognizer is a CRNN that is able to recognize the handwriting that is displayed in the images.

3.3 Training Setup

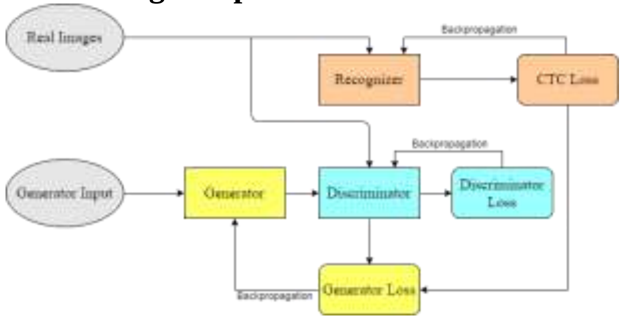

Fig. 5. Training Setup

The setup for implementing the training is as shown in the above diagram. The discriminator tries to distinguish between the fake samples generated by the generator and the real samples from the dataset. These predictions give rise to a discriminator loss that is used to train the discriminator network. The recognizer is trained solely on the real data and generates a CTC loss. Both the CTC and discriminator losses are used to compute a generator loss that is used to train the generative network.

We use the Hinge Loss to compute the discriminator and generator losses.

$$L_D = \mathbb{E}_{(x,s)\sim p_{data}}[\min(0, -1 + D(x))] - \mathbb{E}_{z\sim p_z, s\sim p_w}[\min(0, -1 - D(G(z, \varphi(s))))]$$

$$L_{(G,\varphi(s))} = -\mathbb{E}_{z\sim p_z, s\sim p_w}[D(G(z, \varphi(s)))] + \mathbb{E}_{z\sim p_z, s\sim p_w}[CTC(s, R(G(z, \varphi(s))))]$$

$$L_R = +\mathbb{E}_{(x,s)\sim p_{data}}[CTC(s, R(x))]$$

4 Results

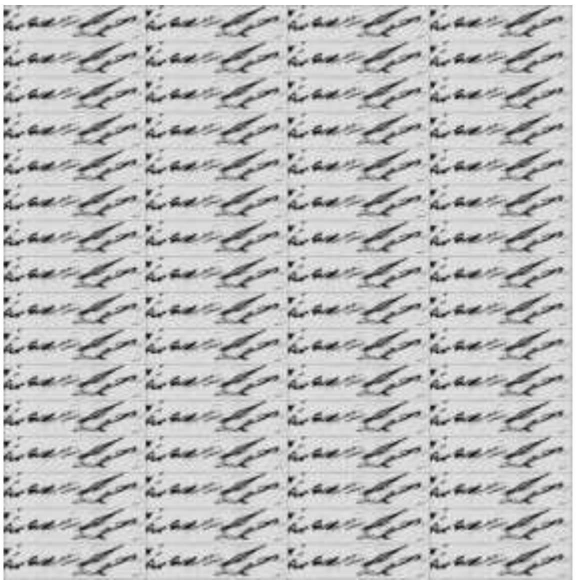

Fig. 6. Model output for input word "amit"

As is visible from the generated results, the model is able to learn a good representation of the handwritten images. It starts to generate images that bear the likeness of handwriting resembling the dataset.
Although lacking a readable nature, we believe this is a great start to solving this problem, and the model can be further bettered by making apt modifications.

5 Conclusion

In conclusion, we have successfully created a neural network that can learn the underlying distribution of the data provided to it in an unsupervised setting. A language model is used to convert the words into a proper representation for the network which is fed into the generator. The architecture, principles, and training of a complex unsupervised model were understood. In the future, we would like to optimize the model further to improve the generation quality.

6 Future Scope

We would like to improve the performance of the model by exploring several other training paradigms (like self-supervised learning) for this task. We would also like to explore other approaches to the generator architecture (like in Fogel et al[16]).

7 Acknowledgement

We would like to thank our supervisor for her support and guidance. We would also like to thank the HOD of the Department of Computer Engineering, RAIT for this opportunity. We would also like to thank all the people whose contributions have led up to this research.

References

1. Adversarial Generation of Handwritten Text Images Conditioned on Sequences Eloi Alonso et al - https://arxiv.org/pdf/1903.00277.pdf
2. Generative Adversarial Networks - https://arxiv.org/pdf/1406.2661.pdf
3. NIPS 2016 Tutorial: Generative Adversarial Networks - https://arxiv.org/pdf/1701.00160.pdf
4. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks - https://arxiv.org/pdf/1511.06434.pdf
5. Improved Techniques for Training GANs - https://arxiv.org/pdf/1606.03498.pdf
6. Modulating early visual processing by language by Harm de Vries et al - https://arxiv.org/pdf/1707.00683.pdf
7. Large Scale GAN Training for High Fidelity Natural Image Synthesis by Andrew Brock et al - https://arxiv.org/pdf/1809.11096.pdf
8. Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition by Theodore Bluche and Ronaldo Messina - http://www.a2ialab.com/lib/exe/fetch.php?media =public ations:bluche_icdar2017_gates.pdf
9. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks by Alex Graves et al - https://www.cs.toronto.edu/~graves/icml_2006.pdf
10. Spectral Normalization for Generative Adversarial Networks by Takeru Miyato et al - https://arxiv.org/pdf/1802.05957.pdf
11. Self-Attention Generative Adversarial Networks by Han Zhang et al - https://arxiv.org/pdf/1805.08318.pdf
12. Long Short Term Memory - https://dl.acm.org/doi/10.1162/neco.1997.9.8.1735
13. Sequence Transduction with Recurrent Neural Networks - https://arxiv.org/pdf/1211.3711.pdf%20http://arxiv.org/a bs/1211.3711.pdf
14. Generating Sequences with Recurrent Neural Networks -

https://arxiv.org/pdf/1308.0850)

15. Synthesizing and Imitating Handwriting Using Deep Recurrent Neural Networks and Mixture Density Networks - https://ieeexplore.ieee.org/document/8493843

16. ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation - https://arxiv.org/abs/2003.10557