Application of Long Short-Term Networks in Smoothing Handwriting

YuMei Bennett, Edward Fry, Muchigi Kimari, Ikenna Nwaogu, and John Santerre

Master of Science in Data Science, Southern Methodist University,  
Dallas, TX 75275 USA

{ybennett, edwardf, mkimari, inwaogu, jsanterre}@smu.edu

Abstract. Dysgraphia is a mental disorder that affects fine motor skills such as writing. In this paper, recent development in neural network deep learning methods are presented in order to help smooth and style one’s handwriting. Our work focuses on Long Short-Term Memory (LSTM) networks. Application is developed to take input of text string, to generate handwriting in a choice of style; or take one’s handwriting input, smooth it out to be more legible without losing writer’s handwriting style.

Keywords: Dysgraphia, LSTM. Long Short Term Memory, Generative Adversarial Networks, GAN, Handwriting, Smooth Poor Handwriting, Style Transfer, Object Detection

1. Introduction

Although handwriting applications have been made available on smart phones, tablets and PC screens, these note-taking or form-filling applications generally rely on the electronic stylus to provide letter by letter stroke information. In conjunction with language dictionary, these apps were able to provide acceptable accuracy for general usage. Handwriting recognition or handwritten text recognition (HTR) is still considered a challenging problem to solve. The high variance in handwriting styles between people along with the poor quality of the handwritten text compared to printed text pose significant hurdles in converting it to machine readable text.

This project is aimed to help people with dysgraphia, a developmental deficit in the acquisition of writing skills (developmental dysgraphia), to gain confidence and encourage creative development without having to worry about showing their own handwriting at any time during interaction with others. Dysgraphia is common and could have significant consequences for those who suffer from them. Dysgraphia impacts both child and adult, and there is no cure for the disease. Dysgraphia is not life threatening, but it is something that has been misunderstood and that has received relatively little attention from researchers.

In this project, we take on the challenge not only for normal handwriting recognition but take a step further to style dysgraphia handwriting. Recent advancements in deep learning such as the advent of transformer architectures have fast-tracked our progress in cracking handwritten text recognition. Recognizing handwritten text is termed Intelligent Character Recognition (ICR) because the algorithms needed to solve ICR need much more intelligence than solving generic machine printing text recognition.

1. Literature Review

2.1 What is Dysgraphia?

Dysgraphia is a disability that affects fine motor skills like writing, buttoning a shirt, or tying a shoelace [30]. It’s a neurological disorder that can affect children or adults. People with dysgraphia tend to find it difficult to express their ideas on paper. In addition to difficulty in expressing their ideas, academic progress for children with dysgraphia can be vastly affected [25].

Below is an example of handwriting from a child with dysgraphia.



Figure 1. Dysgraphia Handwriting Example

Most children with dysgraphia are bright (and are often skilled readers and speakers) [31]; therefore, it is a dilemma of our society that in most of the cases when parents find their child confronting written impairment, they consider it as if he or she is reluctant towards studying and learning. This means that a child with dysgraphia could easily reach adulthood without receiving any diagnosis, constantly trying to self-correct, feeling frustrated, and having low self-esteem.

Since so many adults with dysgraphia remain undiagnosed, it’s difficult to estimate just how many are living with the condition. In children, the rate is often estimated between 4 to 20 percent. Since dysgraphia can’t be outgrown, many adults are still living with this learning disability [29].

2.2 Importance of Legible Handwriting

Comprehensible handwriting is important when it comes to academics. Writing plays a pivotal role in the development of any child [30]. It makes a student capable of transferring the attained knowledge, honing cognitive skills, augmenting the confidence level, and, above all, legible handwriting skills that anticipate success in the future.

If writing has been a lifelong challenge, a patient is likely to have devised strategies to compensate or to avoid writing altogether. With that in mind, symptoms of dysgraphia in adults will manifest as more than just indecipherable handwriting; they’ll also appear in the purposeful avoidance of writing and in weak fine motor skills. Symptoms of dysgraphia at home might look like [32]:

* Highly illegible handwriting, often to the point that even the patient can’t read what they wrote
* Difficulty drawing, tracing, or painting
* Avoids writing whenever possible; prefers a digital grocery list to a written one, for instance
* Makes spelling errors in simple notes
* Inappropriate sizing and spacing of letters
* Unusual body or hand position when writing
* Tight hold on pen or pencil resulting in hand cramps
* Omitting letters and words from sentences

As more jobs depend on computers, writing may not factor into someone’s day-to-day work. Dysgraphia can still cause challenges by making other fine motor tasks difficult. Symptoms of dysgraphia in the work environment may include [33]:

* Trouble filling in routine forms by hand, particularly if they require fitting words into set boxes
* Illegible handwriting: for instance, not being able to read one’s own meeting notes or coworkers complain that memos are indecipherable
* Mixes lowercase and uppercase letters, or print and cursive letters, seemingly randomly
* Often leaves out individual letters or the ends of words, particularly when writing quickly
* Has trouble telling when words are misspelled
* Often uses grammatically incorrect sentences in emails or reports
* May be overly reliant on simple sentence structures
* Prefers to give or get directions orally instead of in writing
* Has trouble “getting to the point” in written communication; emails may be rambling, or reports may repeat the same ideas several times
* Able to explain self clearly when speaking but not when writing

People with dysgraphia often struggle to multitask when writing. Their difficulty in physically writing down each word can lead to missed parts of what is said during meetings or lectures.

Students with dysgraphia may also be accused of being careless because their handwriting isn’t legible. This can affect self-esteem and lead to anxiety, a lack of confidence, and negative attitudes toward school.

2.3 What causes dysgraphia?

In childhood, dysgraphia is usually the result of a problem with orthographic coding [29]. Orthographic coding is an aspect of working memory that allows you to permanently remember written words and the way your hands or fingers must move to write those words.

Researchers are still learning the reasons why some children have learning disabilities, such as dysgraphia. Learning disabilities can be hereditary or can be related to abnormal prenatal development [31].

2.4 Does keyboarding solve all the problems?

Keyboarding does help children to express their ideas at the level of their intellect, and it should be encouraged; however, handwriting should not be ignored completely for several reasons [33]:

* There are times when handwriting is needed, e.g. for certain school subjects (such as math and science) and in other everyday circumstances.
* There is some evidence that the physical act of handwriting helps the flow of ideas for written composition in ways which keyboarding doesn’t.
* Handwriting is very personal. It is an expression of a person’s identity. In adolescence it is common to find young people adopting a style of handwriting to suit their personal image.

2.5 Treatment for Dysgraphia

There is no cure for dysgraphia. Occupational therapy may be helpful in improving handwriting skills. Therapeutic activities may include:

* Holding a pencil or pen in a new way to make writing easier
* Working with modeling clay
* Tracing letters in shaving cream on a desk
* Drawing lines within mazes
* Doing connect-the-dots puzzles

There are not many studies about dysgraphia, so the prevalence of this writing disability is not known; however, it is estimated that between 5-20% of the population has a varied degree of dysgraphia. Since people with dysgraphia are intelligent and high functioning, many misunderstand their disorder as a quirk rather than a problem. Dysgraphia is real and can have a significant negative impact.

However, recent developments in machine learning may present a solution. In this paper, we explore using new algorithms, such as Generative Adversarial Networks, that may be able to take unreadable handwriting and present it in a realistic handwritten text. First, sample handwriting is used to train the models of interest. Then, we analyze the results of that training on new samples. The results section describes our models and methods in greater detail. Ethical considerations of changing handwriting are then discussed, followed by conclusions and recommendations for follow-up work.

2.6 History of Deep Learning and Neuron Networks

While not commonly known today, deep learning has been around since the 1940s [34].

A broader look at the history of Deep Learning reveals 3 major waves of advancements:

* Cybernetics — During 1940–1960
* Connectionism — During 1980–1990
* Deep Learning — Since 2006

Cybernetics was kicked off by the development of the McCulloch-Pitts Neuron. It was an attempt to mimic the biological neuron. It was based on a linear model that would take various inputs, for each input the model had some weights. This model could only output True/False based on the inputs and manually set weights.

Connectionism, also known as Parallel Distributed Processing, become popular in the 1980s, and this approach was inspired by cognitive sciences. The concept of Artificial Neural Network (ANNs) was introduced during this wave. The main idea behind ANNs was to develop a network of individual units that can be programmed to achieve intelligent behavior. This was the first time the concept of hidden layers was introduced [34].

This approach is very similar to how the human nervous system works. During the wave of Connectionism, many models, like LSTM, Distributed Representation and Processing, and Back Propagation to train deep neural nets, were developed and continue to remain key components of various advanced applications of Deep Learning to this date.

Unfortunately, the over hype of artificial intelligence has sent AI industry into a winter season. Until 2006, a breakthrough was made by Geoffrey Hinton’s Greedy Layer-wise Training to train Deep Belief Networks. In the simplest of forms the DBN’s are a composition of multiple hidden layers with each layer containing various latent variables. Connections exist between layers but not between the variables inside each layer [34].

The advancements by Geoffrey Hinton were used by other researchers to train different types of Deep Networks. This enabled researchers around the world to train deeper and deeper neural networks and led to the popularization of the term Deep Learning.

Significant advancement in computer chipsets has also fueled the recent artificial intelligence development. Some algorithms developed in the past started to give better results when trained on larger and larger datasets. The better results attracted more researchers to look for better and more efficient models.

2.7 Project Objectives

This project seeks to produce a handwriting smoother that can be used by two audiences. The first type of audience would be normal (non-data scientist) people, who would simply scan a sample of their handwriting into an application and have smoothed results returned. The second audience would be the machine learning community, who would have access to the project code (open source with an MIT license) and to a full software package that can be imported and used in code.

To meet these goals, the smoother will need to be a trained machine learning model, so the next step is to evaluate possible candidates through the literature research and expert guidance. Five potential machine learning options emerge: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Generative Adversarial Networks (GAN), Neural Style Transfer (ST), and Long Short-Term Memory (LSTM). Of these, GAN and LSTM seem to show the most promise, to be through, the project investigated all the options.

Convolutional Neural Network:

A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers, and normalization layers. The layers of neurons are arranged in such a way as to cover the entire visual field, thus avoiding the piecemeal image processing problem of traditional neural networks [15].

Images typically have multiple channels (for example, Red-Green-Blue channels). It is possible to modify the traditional CNN architecture to work with multiple channel inputs. This removes limitations and increases the efficiency for image processing, resulting in a system that is far more effective and simpler to train for image and natural language processing.

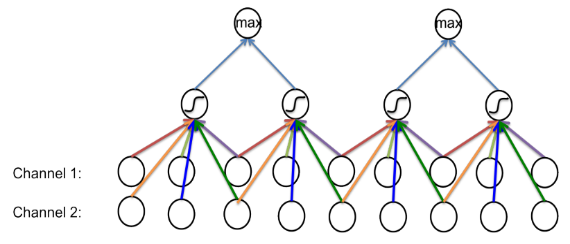


Figure-2 Image with Two Channels CNN

Recurrent Neural Network:

Recurrent neural networks (RNN) are a class of neural networks that allow previous outputs to be used as inputs while having hidden states. RNNs are a rich class of dynamic models that have been used to generate sequences in domains as diverse as music, text, and motion capture data. RNNs can be trained for sequence generation by processing real data sequences one step at a time and predicting what comes next [14].

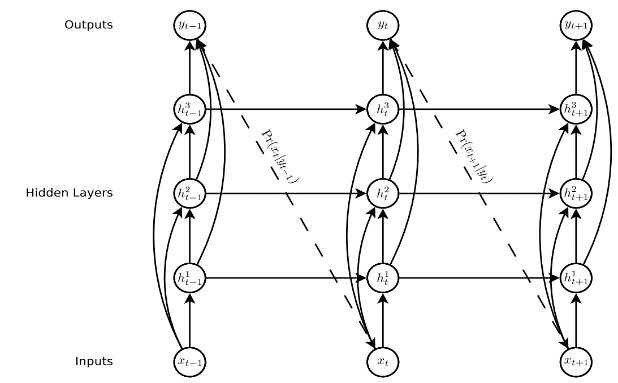


Figure-3 Deep Recurrent Neural Network [14]

Neural Style Transfer:

This is an optimization technique used to copy the style from the style image and apply it to the content image [26]. “Style” is an indication of the patterns, the brushstrokes, and so on of handwriting. Neural style transfer takes three images: a content image, a style reference image, and an input image to be styled and then blended together such that the input image is transformed to look like the content image but “painted” in the style of the style image.

The principle of neural style transfer is to define two distance functions, one that describes how different the content of two images are and one that describes the difference between the two images in terms of their style. Then, given three images, a desired style image, a desired content image, and the input image (initialized with the content image), the input image is transformed to minimize the content distance with the content image and its style distance with the style image.

A 2016 paper by Johnson et al. was the first to train an independent neural network to stylize images in a single, feed-forward pass. Large VGG16 models pre-trained on ImageNet are used for feature extractors, and a relatively small encoder-decoder network serves as the transfer network. In this approach, a single transfer network is trained for each desired style.

In 2017, a year after the original fast style transfer technique was published, researchers at Google extended the technique to allow a single transfer network to produce images in multiple styles and even blend more than one style together [36]. Their main contribution was the inclusion of “conditional instance normalization” layers within the network so that the stylized image produced could be conditioned on an additional model input.

Generative Adversarial Networks:

This is a Neural Network used for unsupervised machine learning. Developed by Ian Goodfellow in 2014, GANs can be simplified into two parts, the generator and the discriminator. The main idea is that the generator creates fake data, and the discriminator validates to see if its real or fake. The goal for the generator is to fool the discriminator [3]. Since GANs are typically used in processing many images, it can be applied to improving bad handwriting.

In a paper published by Shengyu Zhao et al. [2], Differentiable Augmentation (DiffAugment) is proposed to further enhance traditional GAN, a simple method that improves the data efficiency of GANs by imposing various types of differentiable augmentations on both real and fake samples. DiffAugment enables model to adopt the differentiable augmentation for the generated samples, effectively stabilizes training, and leads to better convergence. Experiments demonstrate consistent gains over a variety of GAN architectures and loss functions for both unconditional and class-conditional generation.

GAN is a relatively new deep learning methodology, and multiple adversarial loss functions for training either generative or discriminative models exist. Yet it remains unclear what certain types of functions are valid adversarial losses and how these loss functions perform against one another. A paper by Yi-Hsuan Yang etc. in May 2020 [8] proposes a deeper understanding of adversarial losses by decoupling the effects of their component functions and regularization terms. A simple comparative framework, dubbed DANTest, is used to systematically compare different adversarial losses.

Long Short Term Memory (LSTM):

LSTM (Long Short Term Memory) is an enhanced version of RNN (Recurrent Neural Network). LSTMs are often referred to as fancy RNNs. Vanilla RNNs do not have a cell state. They only have hidden states and those hidden states serve as the memory for RNNs.

It can be difficult to train standard RNN to solve problems that require learning long-term temporal dependencies. This is because the gradient of the loss function decays exponentially with time (called the vanishing gradient problem).

LSTM has both cell states and a hidden state. The cell state can remove or add information to the cell, regulated by "gates". And because of this "cell", in theory, LSTM should be able to handle the long-term dependency. LSTM solves the problem of vanishing and exploding gradients during backpropagations. This is achieved by using the memory cell.

When we move from RNN to LSTM, we are introducing more & more controlling knobs, which control the flow and mixing of Inputs as per trained weights. And thus, bringing in more flexibility in controlling the outputs.

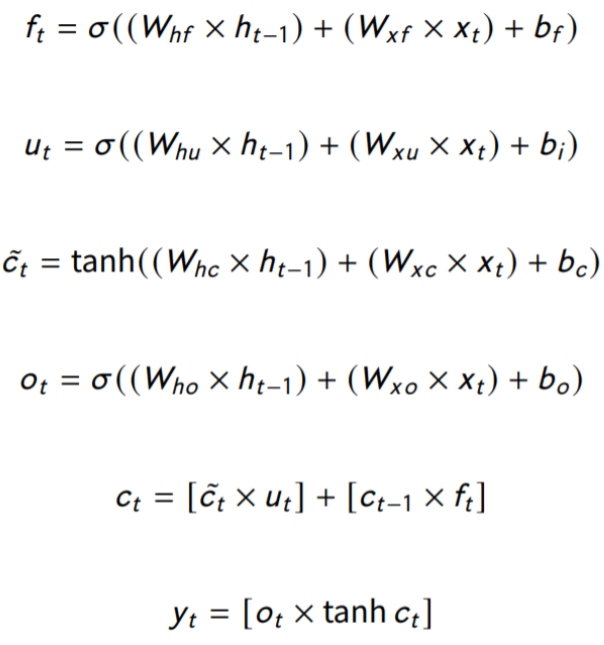
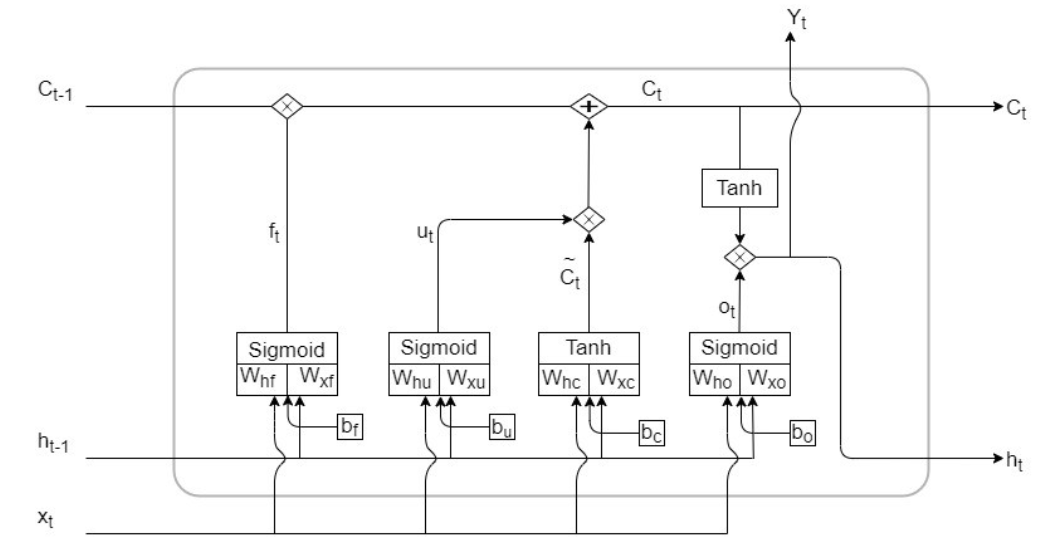


Figure-4 LSTM Cell Architecture [38]

1. Evaluating the Best Approach

Initial requirements for the system were obtained through interviews with Dr John Santerre. The final product is to be a command-line utility that takes as its input a data representation of human handwriting and gives as its output a smoothed version of that handwriting in the same data format as the input. The application should also graphically display the original handwriting and its smoothed output.

A total of 8 sources for the data were considered.

1. *UCI Letter Recognition Data Set* [17] – This repository is maintained by the University of California at Irvine and includes about 20000 capital letters in 20 different fonts.
2. *The Chars74K dataset* [18] – This collection of data includes about 74000 letters and numbers, but only 3410 of them are handwritten.
3. *EMNIST Dataset* [19] – These handwritten characters have been converted to the same 28x28 structure as the famous MNIST dataset. It includes 145,600 letters and 280,000 digits.
4. *MNIST DATABASE of handwritten digits* [20] – This famous dataset includes 70,000 handwritten numeric digits in 28x28 format.
5. *CVL-DATABASE* [21] – 310 writers wrote 7 texts out by hand in cursive script in this dataset.
6. *IAM Handwriting Top50* [22] – This collection of handwritten passages is broken out by sentences, lines of text, and individual words. It is mostly written in cursive.
7. *Yu Qiao* – This site is a collection of links to various handwriting samples and resources.
8. *Novel dataset creation* – In the event that no existing datasets contain the quality, type, or number of samples needed, a new dataset can be commissioned using a service such as Mechanical Turk.
9. *IAM Online Handwriting Database* [37] – Whereas most of the data sources that we examined were raster images, this data source is a time series of {x,y} data points captured as participants wrote on a digital whiteboard.

Python is the natural programing language of choice for this project. It is widely regarded as a suitable and standard programming language for machine learning and data science. Python is preferred because it supports object-oriented, and it has a wide variety of packages for cleaning data and machine learning. Tensor Flow and its later arrival cousin PyTorch are popular in the Data Science research community. In addition, PyTorch has similar syntax to NumPy. The only difference is an option to use Graphic Processing Units (GPUs) [28]. The use of GPUs in PyTorch speeds up the training of deep learning models.

Images by their very nature often have a lot of features since the individual pixels are the data sample set, a 28x28 black and white image would have 784 features. Because of this, algorithm performance becomes a concern. To address this, we need to implement a distributed Python computing platform, Dask and Spark are on top of the list. Both platforms automatically split data and computation amongst a potentially unlimited number of computing nodes known as clusters. Thus, additional computers can be added to the cluster at any time to increase the speed of processing.

In addition, both Dask and Spark supports the use of graphical processing units (GPUs), which are highly optimized for the sorts of linear algebra operations that most machine learning algorithms employ, thus adding both a greater quantity and a more performant set of computation resources to the execution of these machine learning algorithms.

Several promising techniques were explored. Style transfer, while not an algorithm per se, is ultimately the goal of the project, and the other algorithms were chosen for their potential to transfer writing styles. Generative Adversarial Network (GANs) were promising from their ability to find a desired output from initial random noise. And recurrent neural networks (RNNs) have been examined in other contexts for handwriting recognition. All of which operates on image pixels.

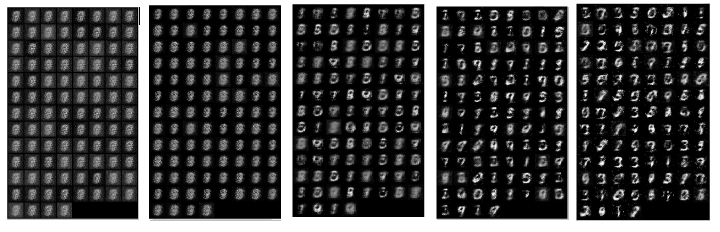
In order to smooth handwriting images, the image had to be recognized correctly by computer program first. Handwriting image recognition isn’t a new topic, there are numerous research and published packages out there, like Textract. This can also be done using the OpenCV package. The application and or model should identify where each line of the handwriting starts and then detect the objects within the line. The line would be a specific size which would contain all the pixels involved in the line where the letters and words are. Both methods were used in this case and performance of both were measured. After separating the letters, each letter will have to be normalized from [0,255] to [0 to 1] before input into the neural network for training.

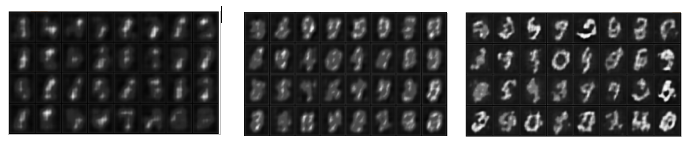
The imaged handwriting recognition although it is considered as mature technology and deployed in varies commercial products, its accuracy was always short from 100% with the assistance of language dictionary input. Applications like smart phone notepad requires manual eye check before commit as records.

The metrics used for measuring the recognition accuracy are Precision and Recall. To be able to apply any of these performance metrics, the measure of the correctness of each detection needs to be established. For any given detection, a bounding box or a circle is placed on the object. The metric that measures the correctness of a giving bounding box or circle is called IoU which is the Intersection over Union. This is the ratio between the intersection and the union of the predicted box or circle and the grounding truth box. Grounding truth box or circle is the correct box representation that would cover each letter. 20 labeled image files from the data sample were used to measure the performance of each of the detection application used and the mean Precision and mean Recall were recorded. Both the openCV application and the pretrained model used had a mean precision of 0.9 and 0.85 respectively and a mean recall of 0.8 and 0.76 respectively. Even though that the pretrained model performed better, both applications were able to detect and separate the letters because whether the letter detected is fully in the designated box or not, it was able to extract part of the data that would be used for modeling and prediction using the GAN model.

The second part is applying Neural network of choice like GAN, or RNN algorithm to train the individual letter detected from the object detection model.

Implementation of GAN and DCGAN (Deep Convolutional GAN) yielded similar outcome. Both models were successful in learning and making predictions from the MNIST data set. Figure-8 below illustrates the simple GAN progression from starting point to end point. The model was configured to use 50 epochs and 100 batches. Figure-9 below shows the DCGAN model learning progression also illustrating from start to finish. The model was configured to use a batch size of 128 and epoch of 5.

Figure-5 Learning Progression for the simple GAN

Figure-6 Learning Progression for the Deep Convolutional GAN

The most promising datasets initially were *EMNIST*, *MNIST*, and *UCI* since they meet the requirements of individual printed numbers and letters in black and white. These are the data sources used in the construction of the training models as show above for GAN and DCGAN.

When apply the same analytic method (OpenCV and GAN/DCGAN) to IAM data set, the result becomes unpredictable, in some cases, the generated styled images are hardly unrecognizable.

It is logical that image based handwriting neural networks have a hard time on handwritings with large variabilities, like Dysgraphia and cursive handwriting. It is operating on pixel position within a defined canvas boundary basis. It is relying on the model to learn all possible styles of handwriting pixel pattern, and each letter and number variable patterns should have enough distance (difference) from other letter and numbers. When input given was a set disassociated and unsimilar handwriting, the model becomes confused. Handwriting unlike a picture, the significant pixels (letter stroke lines) that need to be tracked is only a fraction of the canvas and it is spread out, a picture interested pixels (object blob) typically occupy an area of the picture.

After careful additional study of [14] and [37], it became clear that the advantages of a coordinate-based dataset were significant. The distance between points as given by their capture time gives a very good sense of the stroke. With stroke information, any kind of stroke – printing, cursive, letter-spaces, etc. – could be used, which expands the original proposed capability of the system. In addition, any improvements to the stroke would intrinsically retain the original character of the handwriting. Because of these clear benefits, the IAM online data is explored and utilized as the basis of handwriting smooth application training models.

1. LSTM With IAM Online Dataset

4.1 IAM Online Dataset

The IAM On-Line Handwriting Database (IAM-OnDB) contains forms of handwritten English text acquired on a smart whiteboard [37]. The database contains forms of unconstrained handwritten text, acquired with the E-Beam System. The system collects writer’s pen position (x & y coordinates) data as a set of time series data. The collected data is stored in xml-format, including the writer-id, the transcription and the setting of the recording. The IAM Online Handwriting Database is structured as follows:

* 221 writers contributed samples of their handwriting
* more than 1700 acquired forms
* 13,049 isolated and labeled text lines in on-line and off-line format
* 86,272-word instances from a 11,059 words dictionary

The dataset is organized as group of handwriting segments. Each segment contains multiple short lines of handwriting, under 50 characters including the space. It is organized this way to make it easier to consume as data model training input. An example is show below.

G10-363 ascii text:

People were so kind. I felt like

a shipwrecked mariner who had been

rescued by a luxury liner. Stranger

pressed boxes of chocolates on me.

"Sister, you sure look peaked."

G10-363 stroke position XML:

<StrokeSet>

<Stroke colour="blue" start\_time="15424996.12" end\_time="15424996.55">

<Point x="1507" y="1278" time="15424996.12"/>

<Point x="1500" y="1272" time="15424996.14"/>

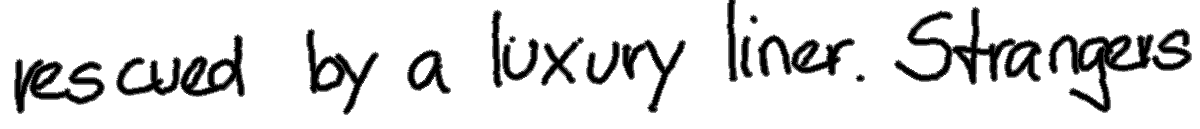
<Point x="1500" y="1272" time="15424996.15"/>

<Point x="1500" y="1272" time="15424996.17"/>

<Point x="1503" y="1275" time="15424996.18"/>

….

G10-363 handwriting image



A few more handwriting images that are part of Online IAM dataset to show the diversity of IAM dataset.

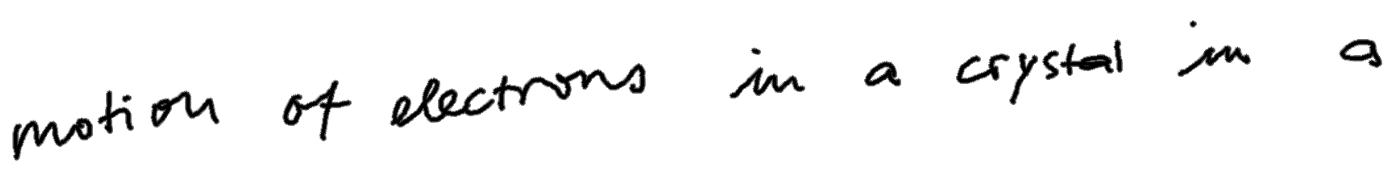












Figure-7 Online IAM Handwriting Style Variation

The Online IAM dataset is divided into a training set, two validation sets and a test set, containing respectively 5364, 1438, 1518 and 3859 handwritten lines taken from 775, 192, 216 and 544 forms input.

4.2 Approach

Ultimately, the method described in [14] utilized a long short-term memory network (LSTM), and that is the direction these efforts eventually followed.

Leverage the principal described in section 5 Handwriting Synthesis of “Generating Sequences with Recurrent Neural Networks”[14], application package is created to allow input ascii text or handwriting in the same format as IAM Online dataset, convincible handwriting is generated.

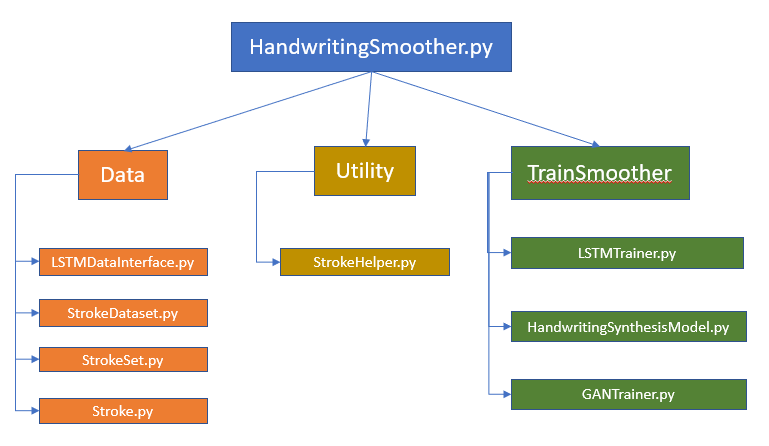


Figure-8 Synthesis Handwriting Application Frame Work

4.3 Data Preparation

Each line was treated as a separate sequence, this means that the possible dependencies between successive lines were ignored. In order to maximize the amount of training data, we used the training set, test set and the larger of the validation sets for training and the smaller validation set for early-stopping. The lack of independent test set means that the recorded results may be somewhat overfit on the validation set; however, the validation results are of secondary importance, since no benchmark results exist, and the main goal was to generate convincing-looking handwriting [37].

4.4 Model Training

The training algorithm uses an Adam optimizer with a learning rate of 0.005. The gradients are clipped to avoid exploding gradients. To help with the computational costs, the program is written to automatically take advantage of CUDA GPUs, if they are available. In this example, CUDA training times were around 15-20% of their times without it (i.e. using only the CPU), which is about an 85% speedup on average. The project code is posted on GitHub EdwardAF-IT/Capstone/blob/master/Code.

For the IAM Online dataset, the code from [14] was first rewritten in Python to validate that the approach in that paper was sound and could be adapted to this project. Then, the code was extended to allow for easy adjustments of the biases, which ultimately determines how much smoothing is applied to the incoming handwriting sample.

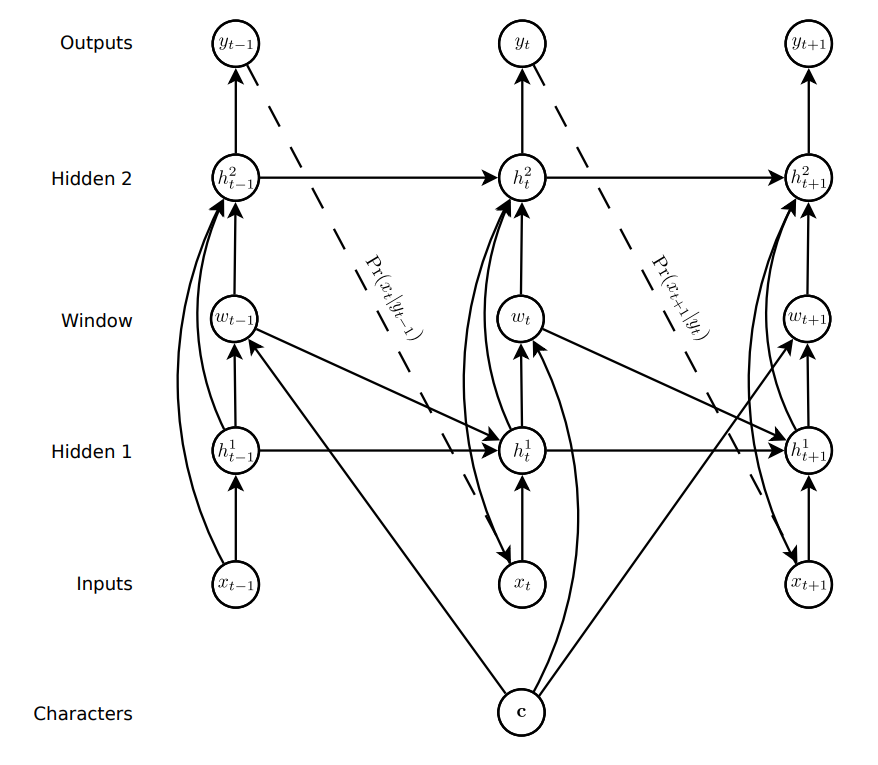


Figure-9 Synthesis Handwriting Network Architecture [14]

In the above adopted LSTM architecture, circles represent layers, solid lines represent connections and dashed lines represent predictions. The topology is like the standard LSTM/RNN network, except that extra input from the character sequence c (ascii text), is presented to the hidden window layer, with a delay in the connection to the first hidden layer to avoid a cycle in the graph.

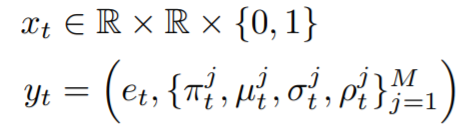
This augmentation allows a prediction network to generate data sequences conditioned on some high-level annotation sequence (a character string, in the case of handwriting synthesis). The resulting sequences are sufficiently convincing that they often cannot be distinguished from real handwriting. Furthermore, this realism is achieved without sacrificing the diversity in writing style learned from training data.

The input layer is size 3, and there are 3 hidden layers with 400 cells in each one. The output layer has 20 bivariate Gaussian mixture components. The loss function uses stochastic gradient descent. Once trained, the same configuration is used to generate or smooth handwriting.

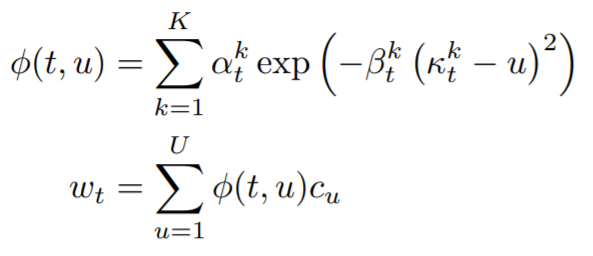
The “soft” window LSTM layer is inserted between hidden 1 and hidden 2, it is convolved with the text string and fed in as an extra input to the prediction network. The parameters of the window are output by the network at the same time as it makes the predictions, so that it dynamically determines an alignment between the text and the pen locations. Put simply, it learns to decide which character to write next.

Each input vector xt consists of a real-valued pair x1, x2 that defines the pen offset from the previous input, along with a binary x3 that has value 1 if the vector ends a stroke (that is, if the pen was lifted off the board before the next vector was recorded) and value 0 otherwise.

A mixture of bivariate Gaussians was used to predict x1 and x2, while a Bernoulli distribution was used for x3. Mixture Densities is a key part of handwriting generation. In simple terms, the network computes a statistical probability function that represents the odds that the next point in the stroke will be at a certain position. The distribution is normalized, which makes it easy to differentiate. Each output vector yt therefore consists of the end of stroke probability e, along with a set of means µj, standard deviations σj, correlations ρj and mixture weights πj for the M mixture components.

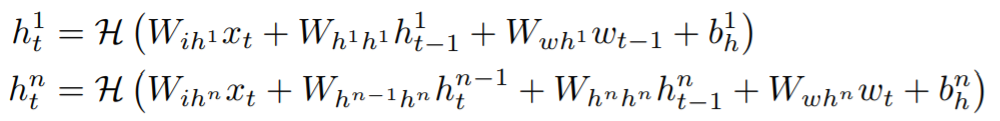


Given a length U character sequence c and a length T data sequence x, the soft window wt into c at timestep t (1 ≤ t ≤ T) is defined by the following discrete convolution with a mixture of K Gaussian functions

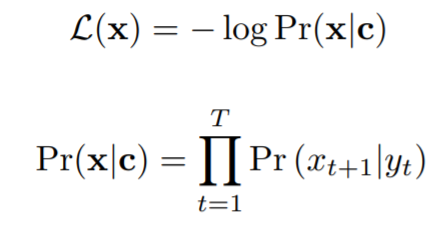


where φ(t, u) is the window weight of cu at timestep t. Intuitively, the κt parameters control the location of the window, the βt parameters control the width of the window and the αt parameters control the importance of the window within the mixture. The size of the soft window vectors is the same as the size of the character vectors cu, assuming a one-hot encoding, this will be the number of characters in the alphabet. Note that the window mixture is not normalized and hence does not determine a probability distribution; however the window weight φ(t, u) can be loosely interpreted as the network’s belief that it is writing character cu at time t.

The wt vectors are passed to the second and third hidden layers at time t, and the first hidden layer at time t+1. The update equations for the hidden layers are



The sequence loss is

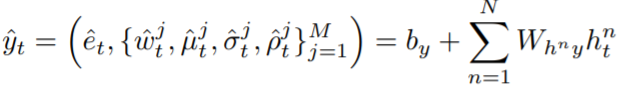


The LSTM Network for training the Handwriting Synthesis model consists of 3 hidden LSTM layers. The middle LSTM layer (soft window) is customized to condition on ascii character input, to restrict the model to write the letter as input character, but with learned handwriting style.

The attention (condition, restriction) mechanism (window layer) is implemented between LSTM1 and LSTM2. LSTM1 takes as inputs the window vectors of the previous time step as well as current stroke coordinates. A dense layer is used taking the output of LSTM1 to compute the parameters of the window vectors. The current window vector is passed on to LSTM1 and LSTM2 as well as the stroke coordinates via skip connections. LSTM soft window and LSTM2 of course take the hidden vectors of the LSTM1 and window LSTM respectively.

The Gaussian mixtures are created using a dense layer. It takes the output of the last LSTM layer LSTM2. This allows the model to scale the output vector size to desired length.

The mean and standard deviation are two dimensional vectors, whereas the component weight, correlation and end-of-stroke probability are scalar. The vectors yt are obtained from the network outputs ŷt where



ŷ is then broken down into the different parameters of the mixture.

* ê is the probability of the end of a stroke given by a Bernoulli distribution
* ω is the weight of each Normal distribution
* 𝜇, 𝜎, and 𝜌 are the mean, standard deviation and correlation factor of each bivariate Normal Distribution.

The following figure shows the alignment implied by the window weights during the training sequence. The model is trained with the neural network consist of 3 layers hidden LSTM layers, 400 LSTM cells per layer, batch size 50, 20 Gaussian mixtures and epoch 30.

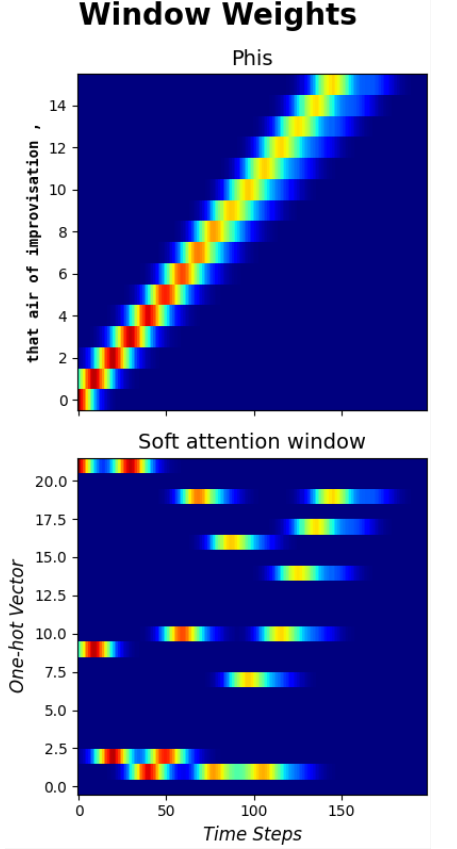
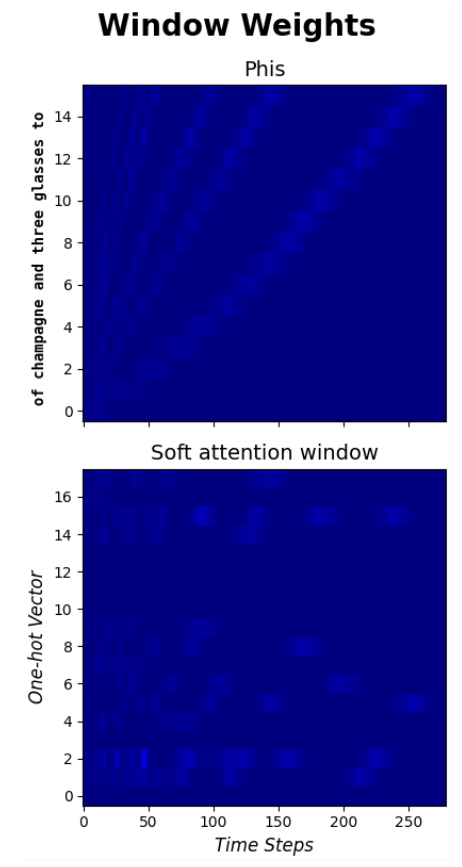
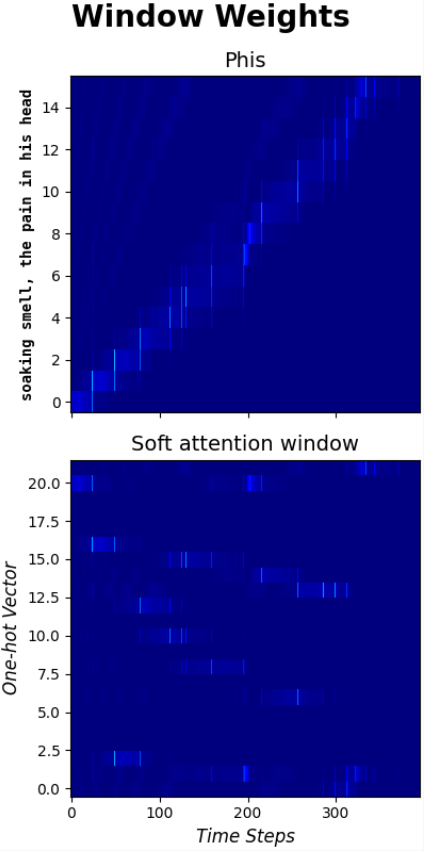
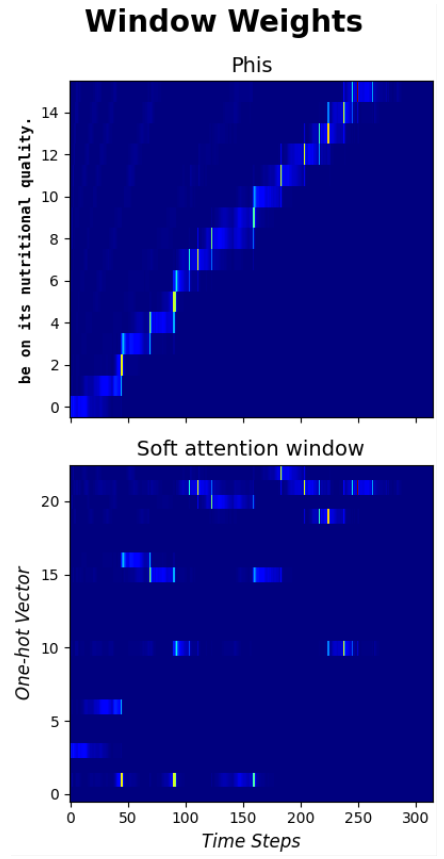
   

Fig. 10 The alignment implied by the window weights during a training sequence

The bright color in first graph shows the beginning of the training sequence when model has not acquired memory of any handwriting sequence. As the training goes on, the model starts to learn and converge.

An interesting thing to note is that the algorithm will continue until it reaches the stopping condition, which is not as trivial as it might seem. Since strokes will nearly always take a variable number of points to construct, it is not a simple matter of enumerating through a for loop. Instead, the algorithm must compare the computed phi to the collection of previous phis. When the computed phi becomes greater than any other value, then the algorithm has reached the end of the sequence and can stop.

Below are examples of a training batch output and Epoch loss figure during the training sequence.

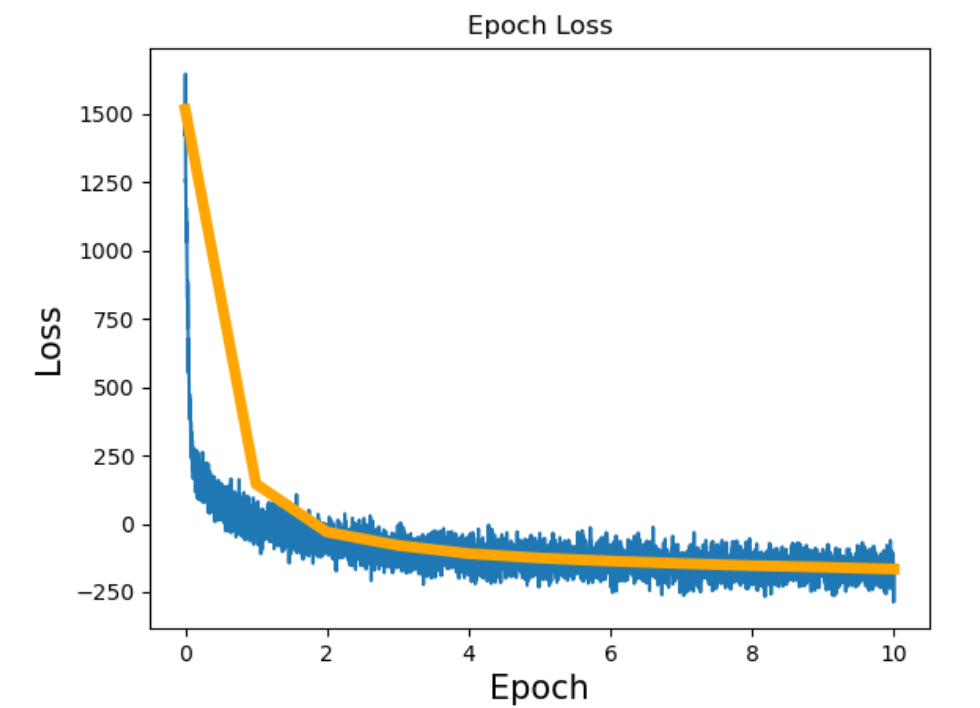








Fig. 11 A training batch run output



Smoothing is very similar to generation. The key difference is that, instead of relying on a text string, the generator takes a sample of existing handwriting. The sample is input to the generation sequence to prime the sequence. This gives the LSTM network a history of data points from which it will predict the next set of points. That lets it maintain the same style as the source handwriting.

4.5 Application

The final product is a command line application written in Python. The application is meant for two type of audiences. For the layman who wants to smooth their handwriting, the program will take a sample of their handwriting as an input along with the preferred model to use from amongst the supported models. Smoothed handwriting will be output.

The second audience of technical professionals can specify training data and one or more models to be trained. The application will train the model(s) and output the computed error of the training.

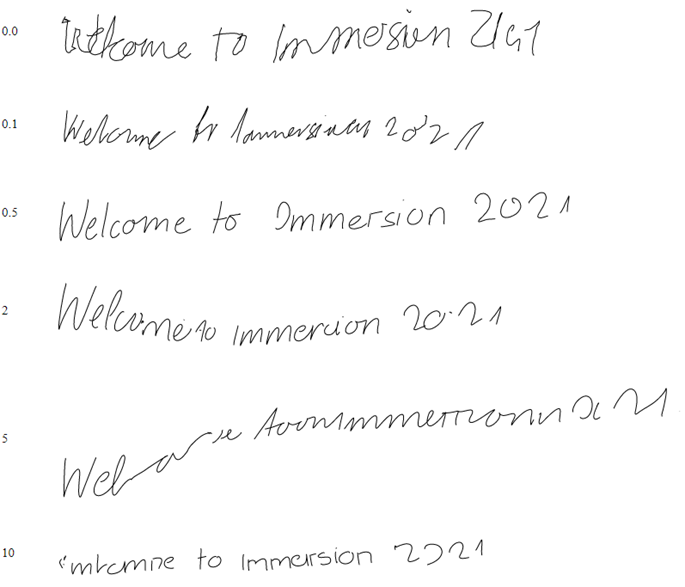
The application can take two formats of input, one is a text string with a limit of 80 characters, the other is one’s handwriting in the same format as the IAM Online dataset that was used to train the model(s). The trained model is then used to generate or smooth the handwriting. Legibility can be controlled with the bias parameter, where higher values (up to 10 or so) will result in greater readability, and low values (to a minimum of 0) give greater variety at the expense of legibility. The result handwriting is saved in SVG format.

The application is written in such a way that it can be executed as a Python script from the command line, as a standalone module, or as an imported Python module used as part of another program.

The main part of the program processes the command line arguments and controls the trainers. Each trainer implements a programming interface that lets them be executed polymorphically by the controller. This allows for a common reporting of test results and training of the models.

4.4 Results

The LSTM model produced impressive results! Below is an example of synthesized handwriting output from the model with different text input. The bias parameter controls how much randomness the model will apply to the output, the lower bias means the model will output significant variations (randomness), higher bias means out put almost looks alike every time.



The resulting style is consistent within itself but distinct from other sequences. In other words, a generated sequence will have the same handwriting style for the entire sequence. However, generating the sequence again, with the same or different character input, will result in a different handwriting style.

At the conclusion of this project, the application will be released as open source so that anyone that needs it would be able to have access to it.

5 Discussion

As explained in the introduction, dysgraphia is a learning disability which strictly affects the ability to write properly. This project seeks to provide a better way to help people with this condition to communicate when writing. In this project, an application takes a text or handwriting input, and the application will generate handwriting with a style of choice.

In the course of exploring techniques for this project, we ran across a lot of information about GANs, or Generative Adversarial Networks. It would be interesting to explore using GANs to replicate the generative and smoothing work presented here. We believe such a network could be utilized to convert handwriting from an image into the online x, y format needed for smoothing. It would also be interesting to find a way to minimize stylistic variance when using the bias adjustment technique while still improving the legibility.

The application frame work is very much extensible for any number of additional methods, additional models with different parameter tuning or preference.

6 Conclusion

Due to the impacts of dysgraphia on children and adults, this application is meant to help people who suffer from this condition to communicate better. The application would create handwriting that is more consistent and easier to read from an original image of a person’s handwriting. The application will not only help people with dysgraphia but also hopefully promote additional research into this area with the planned Python package to drive improvement in the years ahead.

Finally, since the core goal of the project is to assist with a disability, there does not seem to be any negative impact to ethics on the surface. But, if the application are optimized enough to be used as the tools for forging signature, notes, letters, that could be damaging and should be controlled at the source.

Acknowledgments. Jacquelyn Cheun, PhD. – Capstone Professor

References

1. Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu and Fan Yang. Multivariate Temporal Convolutional Network: A Deep Neural Networks Approach for Multivariate Time Series Forecasting. Electronics 2019, 8(8), 876; https://doi.org/10.3390/electronics8080876
2. Shengyu Zhao IIIS, Tsinghua University and MIT Zhijian Liu MIT Ji Lin MIT Jun-Yan Zhu Adobe Research Song Han MIT. Differentiable Augmentation for Data-Efficient GAN Training. https://arxiv.org/pdf/2006.10738.pdf
3. Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. https://arxiv.org/pdf/1706.08500.pdf
4. Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros. Berkeley AI Research (BAIR) Laboratory, UC Berkeley. Image-to-Image Translation with Conditional Adversarial Networks. https://arxiv.org/pdf/1611.070f04.pd
5. Andrew Brock Heriot Watt University, Jeff Donahue DeepMind, Karen Simonyan DeepMind. Large Scale Gan Training For High Fidelity Natural Image Synthesis. https://arxiv.org/pdf/1809.11096.pdf
6. Yasin Yazıcı Nanyang Technological University (NTU), Chuan-Sheng Foo Institute for Infocomm Research, Stefan Winkler National University of Singapore (NUS), Kim-Hui Yap Nanyang Technological University (NTU), Georgios Piliouras Singapore University of Technology and Design, Vijay Chandrasekhar Institute for Infocomm Research. Unusual Effectiveness Of Averaging In Gan Training. https://arxiv.org/pdf/1806.04498.pdf
7. Han Zhang, Zizhao Zhang, Augustus Odena, Honglak Lee Google Research. Consistency Regularization For Generative Adversarial Networks. https://arxiv.org/pdf/1910.12027.pdf
8. Hao-Wen Dong · Yi-Hsuan Yang. May 2020. Towards a Deeper Understanding of Adversarial Losses under a Discriminative Adversarial Network Setting. https://arxiv.org/pdf/1901.08753.pdf
9. David Ha, Douglas Eck. A Neural Representation of Sketch Drawings. https://arxiv.org/abs/1704.03477
10. Adrian Rosebrock on August 24, 2020. Handwriting recognition with OpenCV, Keras, and TensorFlow. https://www.pyimagesearch.com/2020/08/24/ocr-handwriting-recognition-with-opencv-keras-and-tensorflow/
11. Omar Mohammed, Gerard Bailly, Damien Pellier, Univ. Grenoble-Alpes. Style Transfer and Extraction for the Handwritten Letters Using Deep Learning. https://arxiv.org/pdf/1812.07103.pdf
12. Bo Chang∗ Qiong Zhang∗ Shenyi Pan Lili Meng, University of British Columbia. Generating Handwritten Chinese Characters using CycleGAN. https://arxiv.org/pdf/1801.08624.pdf
13. Bharath Narasimhan University of Massachusetts Amherst. Calligraphy Style Transfer using Generative Adversarial Networks. https://people.cs.umass.edu/~bnarasimhan/Cast.pdf
14. Alex Graves Department of Computer Science University of Toronto. Generating Sequences with Recurrent Neural Networks. https://arxiv.org/pdf/1308.0850)
15. Quoc V. Le Google Brain, Google Inc. A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks. Oct 2015.
16. Piotr Teterwak Aaron Sarna Dilip Krishnan Aaron Maschinot David Belanger, Ce Liu, William T. Freeman Google Research. Boundless: Generative Adversarial Networks for Image Extension. https://arxiv.org/pdf/1908.07007.pdf
17. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
18. De Campos, T.E., Babu, B.R. and Varma, M. (2009). The Chars74K dataset: Character Recognition in Natural Images [http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/]. Microsoft India
19. Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from http://arxiv.org/abs/1702.05373
20. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.
21. Florian Kleber, Stefan Fiel, Markus Diem and Robert Sablatnig, CVL-Database: An Off-line Database for Writer Retrieval, Writer Identification and Word Spotting, In Proc. of the 12th Int. Conference on Document Analysis and Recognition (ICDAR) 2013, pp. 560-564, 2013.
22. U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002.
23. Edgard Chammas and Chafic Mokel. Handwriting Recognition of Historical Documents with few labeled data. Nov 2018. March 2019. https://arxiv.org/pdf/1811.07768.pdf
24. Eloi Alonso, Ecole des Ponts and Bastien Moysset. Adversarial Generation of Handwritten Text Images Conditioned on Sequences. Nov 2018. https://arxiv.org/pdf/1903.00277.pdf
25. Biotteau, Maëlle et al. “Developmental Coordination Disorder and dysgraphia: Signs and Symptoms, Diagnosis, and Rehabilitation.” Neuropsychiatric disease and treatment 15 (2019): 1873–1885. Web.
26. Y.-L. Chen and C.-T. Hsu. Towards deep style transfer: A content-aware perspective. In Proceedings of the British Machine Vision Conference, 2016.
27. Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. Spark: Cluster Computing with Working Sets. Amplab, University of California at Berkely, 2010.
28. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 33rd Conference on Neural Information Processing Systems, NeurIPS, 2019.
29. Michael McCloskey and Brenda Rapp. Developmental dysgraphia: An Overview and Framework for Research. US National Library of Medicine, National Institutes of Health Search database PMC 2018. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6238209/
30. Ileana Felipe Santos. Importance of Legible Handwriting. Ecademia.edu 2020
31. Akinmosin, Kemi Adejoke. The Effect of Poor Handwriting on the Academic Performance of the Gifted Learning-Disabled Students. Department of Education for the Gifted and TalentedSchool of Special Education Federal College of Education (Special), Oyo. 2018
32. James Roland on December 7, 2018. What Is dysgraphia? https://www.healthline.com/health/what-is-dysgraphia
33. What Does dysgraphia Look Like in Adults? https://www.additudemag.com/dysgraphia-in-adults-recognizing-symptoms-later-in-life
34. Varun Bansal. The Evolution of Deep Learning. Towards Data Science April 2020. https://towardsdatascience.com/the-deep-history-of-deep-learning-3bebeb810fb2.
35. Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. Stanford University 2016
36. Vincent Dumoulin, Jonathon Shlens, Manjunath Kudlur. A Learned Representation for Artistic Style. Cornell University Feb 2017. https://arxiv.org/abs/1610.07629
37. Liwicki, M. and Bunke, H.: IAM-OnDB - an On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard. 8th Intl. Conf. on Document Analysis and Recognition, 2005, Volume 2, pp. 956 – 961. <https://fki.tic.heia-fr.ch/databases>
38. Gursewak Sing. Demystifying LSTM Weights and Bias Dimensions. Mar 4, 2020