Handwriting Recognition, Learning and Generation

Rajat Shah, Shritek Jain, Sridhar Swamy
Department of Computer Science, North Carolina State University, Raleigh, NC 27695
{rshah6, sjain9, snswamy}@ncsu.edu

Abstract

It is important to digitize handwritten documents for efficient processing and storage. This problem is well known as Intelligent Word Recognition (IWR) and has been an area of study for decades. Our work presents an effective method to not only recognize the contents of an handwritten document, but also to generate handwritten documents from typed inputs with writing characteristics that are specific to a author's writing style. Users can train our model by providing their handwritten documents as inputs and subsequently use this model to generate documents written in their handwriting font from text inputs.

1 Introduction

Handwriting data is converted into digitized form by scanning documents containing written text. With the advent of computerization, recognizing handwritten text is becoming an important task. Given a handwriting sample, the text is initially segmented at the character level and the feature-vectors corresponding to each isolated character is extracted, which can help in determining the writing characteristics of a user. Further, these feature-vectors can be used to recognize and generate a writer's handwriting. Therefore, the whole problem can be broadly broken down into the following three sub-tasks:

1. Recognition: Recognizing the text from an image of a handwritten document.
2. Handwriting Learning: Extracting distinctive user specific features from a user's handwriting.
3. Generation: Generate handwritten document (an image) from text input in the user's own writing style.

2 Background and Related Work

Studying handwriting is a challenging task as it comprises of several distinct sub-tasks. From the point of view of handwriting recognition, there can exist two variants of systems: Online and Offline. Online recognition systems make use of input device movements using sensors to capture a dynamic representation of the handwriting process [1]. Online recognition systems are usually found in hand-held computers such as PDA's and tend to be more accurate than Offline systems as they possess more information about inputs.

Neural Network based pattern recognizers have been shown to achieve human-competitive performances. Convolutional Neural Networks (CNNs) in particular, tend to perform well in handwriting recognition tasks. Several computational models have been proposed that relate force and velocity to handwriting trajectory. Li et al. employed the delta log-normal model to represent the velocity of handwriting trajectory and to encode it as a set of parameters. Bezine et al. [7] proposed a beta-elliptic model to estimate the correlation between geometry and kinematics in fast handwriting generation [8]. Recent advances in handwriting generation have incorporated deep learning techniques such as Recurrent Neural Networks (RNNs) which are designed to output sequences of characters. RNNs can be used to generate complex sequences by predicting one data point at a time. Long Short-Term Memory (LSTM), which is a RNN architecture has given state-of-the-art results in handwriting recognition [3].

3 Methods

As discussed before, our work mainly comprises of 3 main stages: Recognition, Learning and Generation. The methods employed in each of these stages can be described as follows:

3.1 Recognition

It is important to perform several document analysis operations prior to recognizing the text from handwritten documents. These usually comprise of preprocessing activities such as converting an image to gray-scale, thresholding, extracting foreground textual matter by eliminating noise, word segmentation, character segmentation and so on, in order to reduce the complexity of the input. The activities we have carried out in the handwriting recognition phase can be described as follows:

3.1.1 Converting to Grayscale

Color comprises of three components - luminance, hue and saturation. Luminance is by far more important in determining the visual features of an image. We therefore, converted each image to grayscale as it eliminates the hue and the saturation information and only retains the luminance information. Further, grayscaling also reduced the complexity involved in processing an image.

3.1.2 Thresholding

It is one of the simplest segmentation methods. It helped us to separate out the foreground pixels of the image we wanted to analyze. This separation is based on the variation of intensity value with respect to a threshold. If the intensity value of a pixel was greater than a threshold value, we set the pixel value to be 255. If it was less than the threshold value, we set it to 0. Applying thresholding to a grayscale image transformed it into a binary image.

3.1.3 Smoothing

Smoothing is primarily used to reduce or eliminate noise. For our project purposes, we used the median filter for smoothing. Median filter is particularly useful for eliminating salt and pepper noise. Smoothing using median filter can be used to eliminate the noise induced during scanning the documents in the form of grain and so on. It replaces each pixel with the median value of its neighborhood pixel.

3.1.4 Dilation and Erosion

Dilation and erosion were applied primarily to join the gaps resulting from applying the previous preprocessing steps.

Erosion: Erosion erodes away the boundaries of the foreground object. The value of the output pixel is the minimum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is 0, the output pixel is set to 0.

Dilation: Dilation was used to increase the size of the foreground object. Erosion tends to shrink image size, so we recovered it by dilation. The value of the output pixel is the maximum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.

3.1.5 Contours

Contours can be explained simply as a curve joining all the continuous points having same color or intensity. The contours are a useful tool for shape analysis and recognition.

3.1.6 Bounding Rectangles

For each contour determined in the previous step, we determined the bounding rectangle which passed through the furthest points in each of the 4 directions. The image was then broken down into character segments based on these bounding rectangles and each of these segments was scaled to a size of 40x30 pixels and outputted as a separate character image.

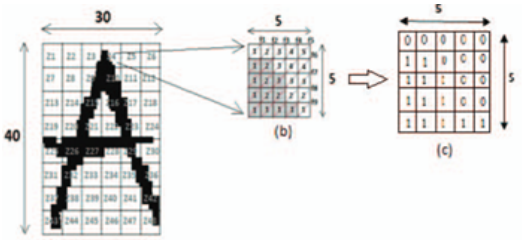
3.2 Handwriting Features Learning

Once we mapped the handwritten document to English text, in order to personalize the handwriting of the writer, we needed to extract user specific features. These user features are key to generating handwritten documents in user's handwriting font. This task is identical to the task of writer identification from a set of handwritten documents. In the following sub-sections, we have described the methods we use for this subtask.

3.2.1 Diagonal-based

In this method, each character image of 40x30 pixels obtained from the preprocessing step was divided into 48 blocks of 5x5 pixels each. We will be referring to these 5x5 pixels as zones hereafter. Each zone comprises of 9 diagonal lines. The foreground pixel values of each diagonal line were summed up to obtain a sub-feature. The resulting 9 sub-features of a zone were averaged to obtain a single feature value. Thus, we obtained one feature from each of the 48 zones, resulting into 48 features of the character image.

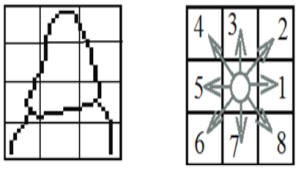
Figure 1: Diagonal-based feature extraction



3.2.2 Direction-based

For direction-based feature extraction, the 40x30 pixel character image is initially subjected to thinning in order to obtain a skeleton of the character by reducing the thickness of each image [4]. Further, the character image is divided into P zones of MxM pixels each. A 3x3 window is labeled from 1 to 8 along the borders in anti-clockwise direction. We move this window over each zone of our 40x30 pixel image. Each image comprises of 12 zones. The center pixel of the 3x3 window is taken as the reference point. If we detect a foreground pixel in any of these 8 directions w.r.t to center pixel, we increment the pixel presence count of that direction. We consider the value obtained for each of the 8 directions to be a feature. Therefore, 8 features are obtained from each zone, resulting into 96 features for each character image.

Figure 2: Diagonal-based feature extraction



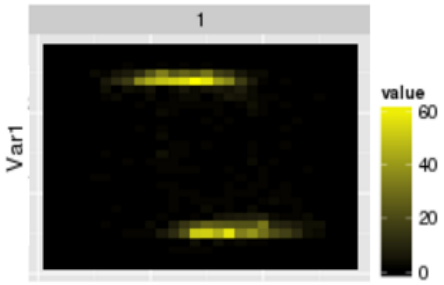
3.2.3 Hybrid

This approach combines the features obtained from the diagonal and direction-based extraction. Combining the features from the previously mentioned approaches gives us 144 features.

3.2.4 Endpoint Distribution

Another important handwriting feature of an author is the end-point distribution which signifies where the author starts and ends the characters most often. This distribution is generated from MNIST Digit dataset and a sample distribution for a single digit is shown in the Figure 3.

Figure 3: Endpoint Distribution for Digit "1"



3.3 Generation

It is important to retain and depict user's writing characteristics such as character glyph, connection style, character spacing and cursiveness for the generated handwriting to look realistic. This necessitates the use of some form of sequence generation algorithm which makes prediction pixel by pixel while also retaining the features representative of an individual's handwriting. The task of handwritten text generation is analogous to sequence generation using Recurrent Neural Networks (RNNs) [3]. RNNs are dynamic models whose utility to generate sequences in diverse domains is proven. RNNs can be trained to generate sequences by processing real data one step at time and making predictions about what would come next in the sequence. RNNs use outputs generated by them as input for future processing which classifies them as dynamic models. RNNs do not use exact templates from the training data to make predictions but they themselves identify a pattern between the training examples and insert it. This dynamic approach is superior to using static models as it would not rely only on training data for pattern matching and prediction. Also, dynamic predictions do not face the curse of dimensionality, hence are much better for sequence generation.

However, there is one major problem associated with large RNNs; they are unable to store previous inputs for long. This reduces the system's stability, the reason being that only recent outputs which were generated by the system itself will be used, giving little opportunity to recover from previous mistakes. To overcome this problem a special architecture of RNNs, Long Short Term Memory (LSTM), is used. LSTM is better at storing and accessing information than standard RNNs.

The computational power of RNNs makes them very hard to train. One of the trained models whose results have been reported itself took around 2 days to train. Hence, experiments involving parameter tuning to evaluate differently trained models seemed infeasible. Once the model is trained, the generation phase simply involves repeatedly predicting one (x, y) pixel point at a time.

4 Plan

The primary goal of our work is to develop an end-to-end system which would generate a customized handwriting font for a user based on the user's writing samples provided as an input to the system. Our primary investigation for this project revolved around determining the features of handwriting styles in order to recognize text from handwriting samples and using those features to generate a customized font, which could be further used to generate handwritten documents from ASCII character inputs. We managed to implement the 3 elements of the project viz. Recognition, Learning and Generation, separately.

For the recognition phase, we used the handwriting samples collected from the students in the class. Our goal for the recognition phase was to identify each character from the handwritten samples collected after segmenting them into word-images. We used the methods described in section 3.1 above for this sub-task and achieved 98.2% accuracy using SVM.

We chose to use an Online handwriting dataset for the generation phase as it is relatively simpler to transcribe raw handwriting data with RNNs because the input forms a 1-dimensional sequence which can be directly fed into a RNN. Further, we chose the Long Short-Term Memory RNN because it possesses the ability to access long-range context. In case of Offline handwriting data, this task becomes much more challenging as the data is no longer than 1-dimensional. As a workaround, we could feed the RNN a single vertical line from the image at a time as a 1-dimensional sequence. However, in this case, the system would no longer possess the capability to handle distortions along the vertical axis. If the image is shifted even by a single pixel along the vertical axis, the image would appear entirely different.

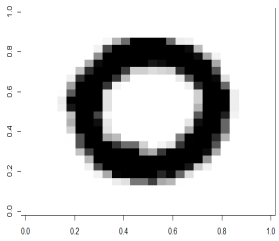
5 Experiments and Results

5.1 Data set

We have used separate datasets for each of the 3 subtasks of the project as follows:

1. For line, word and character segmentation, handwriting data was collected from participants in CSC591 class. This dataset consisted of participants writing on a template consisting of digits, uppercase and lowercase English sentences.
2. For experimenting with the various tasks involved in the recognition phase, we used MNIST Dataset consisting of 70,000 preprocessed images of digits. The data is split into 60% train and 40% test sets. A sample image from the dataset is shown in the Figure 4.

Figure 4: Sample image from MNIST Handwriting Digits Dataset



3. The task of character generation required an Online Handwriting Dataset. We have used IAM On-Line Handwriting Database (IAM-OnDB) containing different forms of handwritten English text acquired on a whiteboard. It consists of handwriting samples from 221 writers. Online handwriting is a good choice for sequence generation due to its low dimensionality (two real numbers per data point) and ease of visualization. A sample snippet from the data is shown in Listing 1.

Listing 1: Sample data from Online Handwriting dataset

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<WhiteboardCaptureSession>
  <WhiteboardDescription>
    <SensorLocation corner="top_left"/>
    <DiagonallyOppositeCoords x="6512" y="1376"/>
    <VerticallyOppositeCoords x="966" y="1376"/>
    <HorizontallyOppositeCoords x="6512" y="787"/>
  </WhiteboardDescription>
  <StrokeSet>
    <Stroke colour="black" start_time="769.05" end_time="769.64">
      <Point x="1073" y="1058" time="769.05"/>
      <Point x="1072" y="1085" time="769.07"/>
      <Point x="1066" y="1117" time="769.08"/>
    </Stroke>
  </StrokeSet>
</WhiteboardCaptureSession>
```

5.1.1 Data Preprocessing

Most of data preprocessing steps required for character segmentation task is reported in section 3.1. These necessary steps are typical of any OCR task. For recognition task, in order to reduce the dimensionality of the character image, Principal Component Analysis (PCA) seems effective. This ensured that the key pixels in the images have proportionate weightage in making predictions.

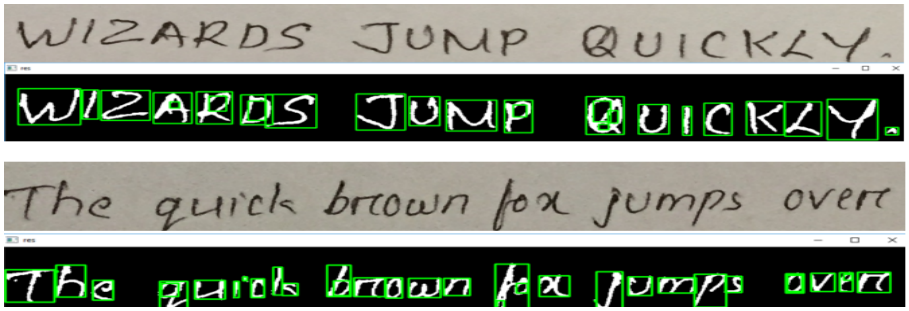
Preprocessing IAM-OnDB dataset required special attention. The raw input data consists of the (x, y) pen co-ordinates and the points corresponding to the action of lifting the pen off the whiteboard. Some recording errors in the (x, y) data was corrected by interpolating to fill in for missing values. After this step, the network is directly trained to predict the (x, y) co-ordinates one point at a time.

5.2 Results and Discussion

5.2.1 Recognition and Learning

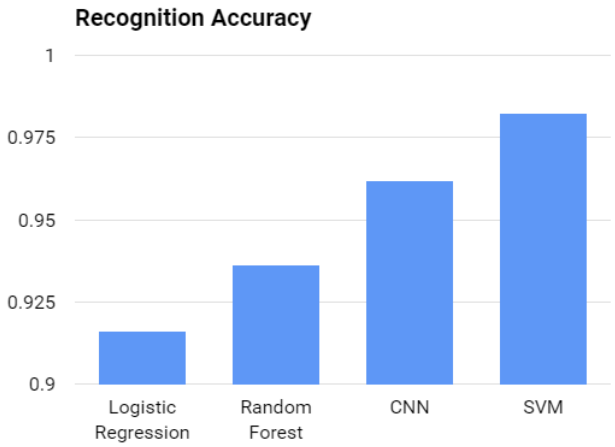
The initial task of character segmentation on an handwritten document is performed on the data collected from the CSC591 class. The result after performing preprocessing activities described in section 3 are shown in Figure 5. As evident, character segmentation has performance challenges due to the connected components (cursive handwriting sample), whereas it has acceptable recognition capabilities when applied to disjoint characters (upper-case handwriting sample).

Figure 5: Input Samples and Character Segmented Outputs



For experimenting with the MNIST dataset and evaluating the effectiveness of character recognition module, we employed 4 Machine Learning methods. Starting with Logistic Regression as the baseline method, **0.916** accuracy is obtained on 28,000 test images. Further, we decided to test our data on Random Forest algorithm as it is known to be quite powerful in multi-class classification. It resulted in an accuracy of **0.93629** with 25 Decision Trees. Convolutional Neural Network (CNN) has been proven to be effective in several Computer Vision tasks. Hence, we tried running a CNN with 1 hidden layer consisting of 50 units on our data which resulted in an improved accuracy of **0.96186**. By increasing units in the hidden layer, the accuracy can probably be further improved but at the cost of increased running time. SVM with Radial Basis Function (RBF) kernel produced the best accuracy of **0.98243** among these methods. The results are summarized in the Figure 6.

Figure 6: Accuracy of Classifiers on MNIST Digit Dataset



5.2.2 Handwriting Generation

Handwriting Generation task when conditioned on character string given as input (and implicitly the starting point of the sequence) is termed as Handwriting Synthesis [3]. The Neural Network Architecture used for Synthesis is illustrated in Figure 7. The best prediction network reported in [3] consists of: 3 hidden layers of 400 LSTM cells each, 20 bivariate Gaussian mixture components at the output later and a size 3 input layer. With one-hot vector encoding of character sequence, the window vectors are of size 57.

RMSPROP algorithm [9] is used for the network training, a variant of stochastic gradient descent where the gradients are divided by a running average of their recent magnitude.

Figure 7: RNN Architecture for Handwriting Synthesis. Circles represent layers, solid lines represent connections and dashed lines represent predictions. Source [3].

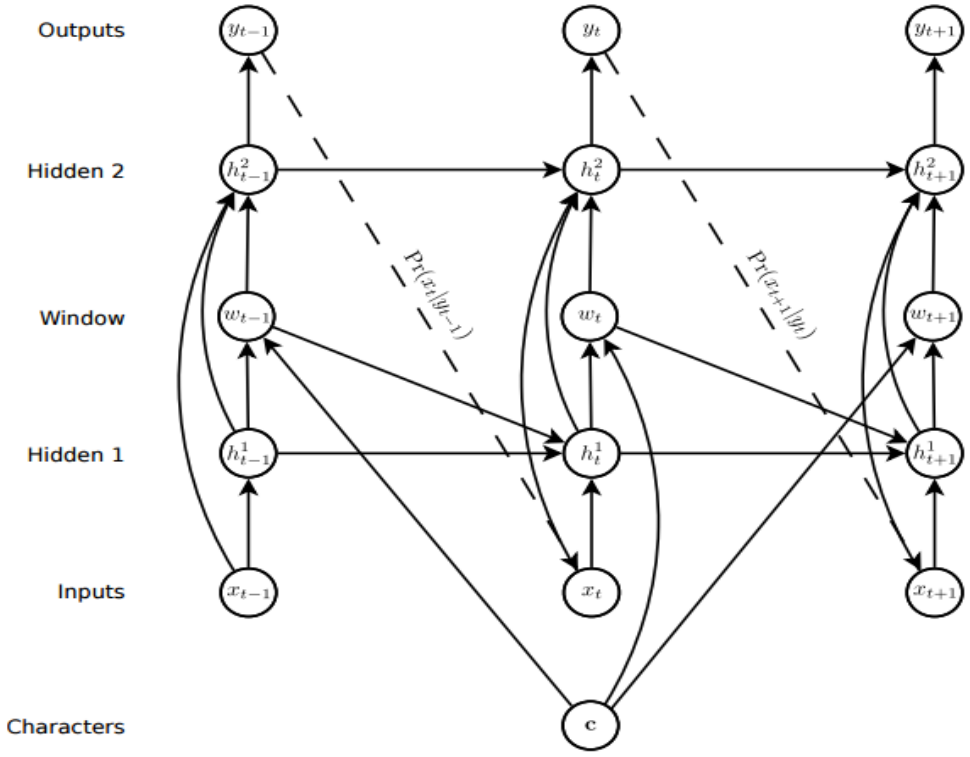


Figure 8: Sample text generated from the trained RNN model

6 Conclusion

In this project, we studied the end-to-end process of customized handwriting font generation using Machine Learning methods. The task of Handwriting recognition has been accomplished and SVM is shown to have best performance (for recognizing digits) among the four classifiers we have used, with an accuracy of 98.2%. Use of more sophisticated Neural Networks can result in better accuracy results, but at the expense of increased training time. For the second

phase of user handwriting feature extraction, we have extracted user specific features from collection of handwritten documents. These features can be validated by using offline handwritten documents from various authors in MNIST dataset. The effectiveness of RNNs for Generation phase can be seen in the Handwriting Synthesis output in Figure 8.

Since the datasets used for each subtasks are different (Offline and Online), we plan to combine these 3 phases and build a complete pipeline in the future work. Moreover, the feature learning phase can be tried on Online dataset which will provide seamless flow across to the next phase of handwriting synthesis. Team work, exposure to new state-of-art technologies and algorithms in the field of Machine Learning and Computer Vision are the learning outcomes of this project.

7 References

[1] Plamondon, Rjean, and Sargur N. Srihari. "Online and off-line handwriting recognition: a comprehensive survey."

[2] Siddiqi, Imran Ahmed, and Nicole Vincent. "Writer identification in handwritten documents."

[3] Graves, Alex. "Generating sequences with recurrent neural networks."

[4] Pradeep, J., E. Srinivasan, and S. Himavathi. "Performance analysis of hybrid feature extraction technique for recognizing English handwritten characters."

[5] Varalakshmi, Aparna, Atul Negi, and Sai Krishna. "DataSet Generation and Feature Extraction for Telugu Hand-Written Recognition."

[6] Li X., Parizeau M., Plamondon R. "Segmentation and reconstruction of on-line handwritten scripts."

[7] H. Bezine, A.M. Alimi, N. Derbel. "Handwriting trajectory movements controlled by a beta-elliptical model."

[8] Zhouchen Lin, Liang Wan. "Style-preserving English handwriting synthesis."

[9] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.