Handwriting Recognition using Convolution Neural Network and Hidden Markov Models

Akruti Kushwaha 201301137
Srivenkata N Mounika Somisetty 201330076

Statistical Methods in AI
Group No. 16

Abstract

Recognizing handwritten text means translating a graphical representation of text into a textual output. Handwriting recognition is used everywhere from everyday things like smart phones to more advanced applications such as signature recognition. This report describes a procedure to develop a handwritten text recognition system based on Convolution Neural Networks and Hidden Markov Models. The application of these two methods in this area has been done previously with good results. The conclusions of this report show the usefulness of these two methods and what they can add to this area of research.

1 Introduction

Offline handwritten word recognition is the conversion of an image into text. Compared to many computer vision problems, the images of text are simple. We can often binarize them, and the task consists of recognizing the shape of the ink. The main challenges involved in the recognition are the segmentation, the different handwriting styles (slope, print, cursive), touching characters, alignment etc. A common technique used to identify a given handwritten input, is to extract characters, and the extract features for each of these characters. With these features, a classifier such as an Artificial Neural Network, Convolution Neural Network, SVM etc is trained to recognize the input character. Following this, the HMM module is used to help improve the accuracy of the prediction. It, in a way, uses the spatial locality of the characters around it to help predict what the word may have been, thus trying to improve the accuracy of prediction of the whole word, and not just the characters. This, however, is entirely dependent on the dataset available and the words in the dictionary of the HMM. As described in this report, we use a trained CNN and HMM to recognize a subset of handwritten words, specifically, due to dataset limitations, printed words written with capital letters.

2 Dataset

An attempt was made to find a dataset with handwritten text segmented till character level, but no such dataset was found. The datasets that were found, were only segmented till word level, and thus would require advanced character segmentation algorithms and the classifier would still yield an output dependent on the accuracy of this character segmentation algorithm. Figure 1 shows a sample from one of the datasets found (IAM). To get good results from images as such would required a lot of pre-processing. Therefore, the dataset finally used was a small dataset made by [1]. The dataset contains 100 examples for every capital letter in the Latin alphabet. The generator used to create the dataset creates random errors in the words given as input. Thus, this provides us a small estimate of actual errors. This dataset is obviously not usable for practical purposes, but it is good enough to test the classifier.

3 Method

Initially, a Convolution Neural Network was trained using the training samples in the above mentioned dataset. Following this, a Hidden Markov Model generator was also trained and the probability matrix required for word prediction was acquired. Next, a pipeline was built to allow any input image to be classified. The pipeline is described in detail below. Initially the image is pre-processed and then sent to the CNN and then, the HMM.

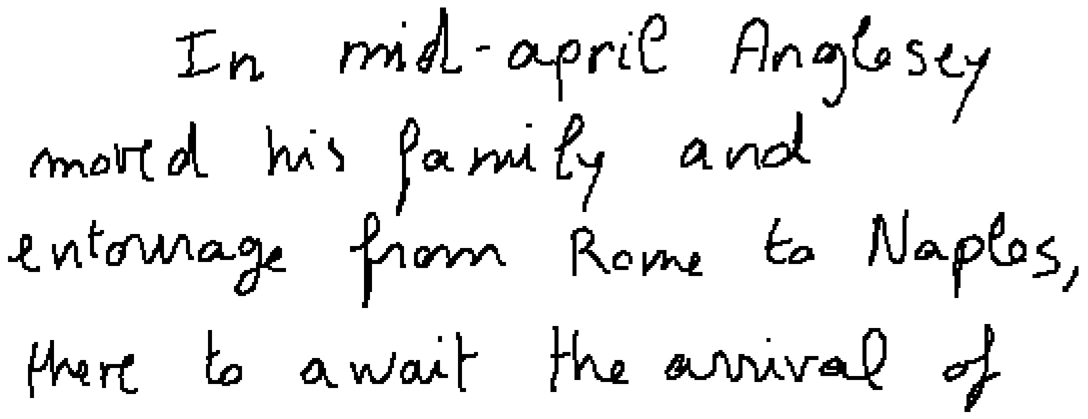
3.1 Pre-processing

The input image of text was initially pre-processed to accommodate multiple lines of text each containing multiple words. This pre-processing was in hopes of making the system invariant to different kinds of inputs. Initially, the images are binarized, dilated (if necessary) and re-sized.

3.1.1 Line Segmentation

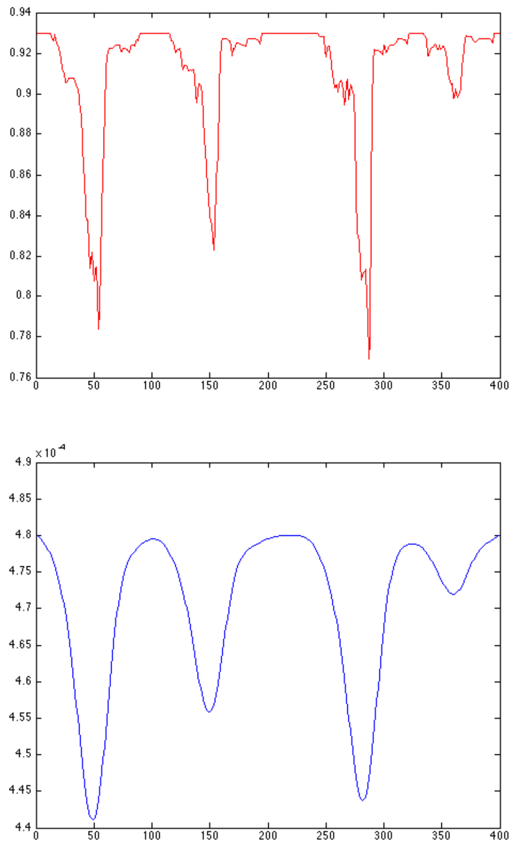
Line segmentation was done using the vertical projection profile method[2].In this method, initially, the input is rotated by 90 degrees and the sum of each column is found. The distinct peaks in the graph plotted correspond to the white spaces between lines while the minima correspond to the lines. The projection function was smoothed using a Gaussian (low pass) filter to get rid of the false local maxima and minima. The local maxima is found using first derivative of the projection function. These maximas indicate the positions of divisions into various lines. This technique is meant to be robust to variations in size of lines.

Figure 1. Sample image from IAM dataset with projection profile given below.



In mid-april Anglesey
moved his family and
entourage from Rome to Naples,
there to await the arrival of

Figure 2. The first image is the raw vertical projection profile, the second is the smoothed counterpart.



3.1.2 Word Segmentation

Each line is converted to the corresponding blob representation as described in detail in [2]. This is achieved by convolving each line with the Laplacian of an anisotropic Gaussian filter. This filter has separate σ values for the x and y direction, allowing us to treat them differently.

$$G_\theta(u, v; \sigma_x, \sigma_y, \theta) = \frac{1}{\sqrt{2\pi\sigma_x}} e^{-\frac{1}{2}\frac{x^2}{{\sigma_x}^2}} * \frac{1}{\sqrt{2\pi\sigma_y}} e^{-\frac{1}{2}\frac{y^2}{{\sigma_y}^2}}$$

The parameters are chosen based on previous analysis. As indicated by the paper [2], the ratio σ_x/σ_y is best between 3-5, $\sigma_y = k * lineheight$. The size of the Gaussian kernel was also experimented with. Each blob is a connected component corresponding to a particular word. Connected component analysis was done on the blobs to segment them.

Figure 3. The various blob images for different kernel sizes

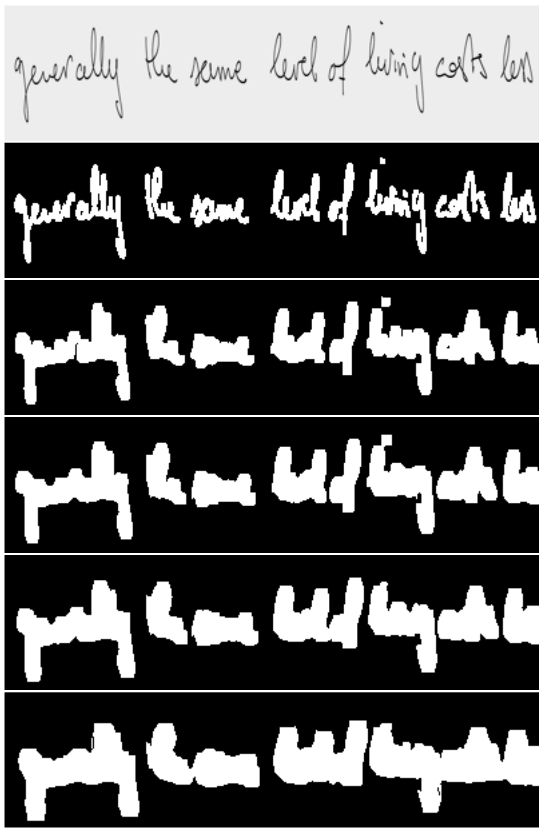
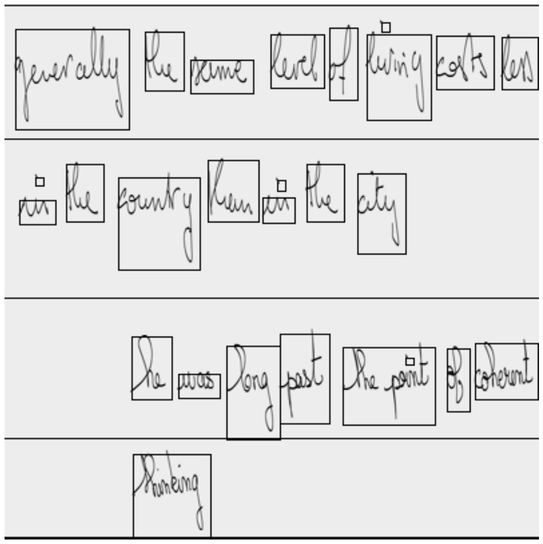


Figure 4. Word-segmented Image


generally the same level of living costs less
in the country than in the city
The was long past the point of coherent
thinking

3.1.3 Slant Correction

Slope correction [3] is first performed on the word images to remove the variability in the slope of writing. The slope of the word is defined as the slope of the writing with respect to the vertical. The underlying assumption used to calculate the slope is that words are normally written with a consistent slope, thus the global slope of the word can be calculated.The dominant slope of the word is found from the slope corrected word which gives the minimum entropy of a vertical projection histogram. Where entropy is defined as,

$$H = -\sum_{j=1}^{N} p_j log p_j$$

The vertical projection histogram is calculated as mentioned above in line segmentation. The distribution is then normalized to have a total area = 1. The basic idea can be demonstrated using a vertical line as an example. When the line is slanted at an angle, it will have a low flat wide distribution whose width is $l \cos \alpha$, where l is the length of the line and α is the is angle of the line to the horizontal axis. When the line is upright, the distribution will now be tall and narrow, which will result in a lower entropy measure than for the low flat distribution of the slanted line. Thus, the entropy gives a measure of the uprightness of the word The vertical projection histogram is calculated by first correcting the binary image by an arbitrary angle i, using sheer transformation.The correction angle, α_m, was found from the minimum entropy. The slope of the grey-scale word image is then corrected using α_m and the resulting in a slope corrected image.

3.1.4 Character Segmentation

Due the the relative uniformity of spaces between characters in the dataset, character segmentation can also be done using the vertical projection profile similar to that done for line segmentation, varying the parameters chosen for the low pass filter. After segmentation, each character is re-sized to $28x28$ size and saved for input to the neural network for classification.

3.2 Convolutional Neural Network

The convolutional neural network (CNN or ConvNet) is a type of feed forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. They were inspired by biological processes and are variations of multilayer perceptrons designed to use minimal amount of preprocessing having wide applications in image and video recognition.

3.2.1 Backpropagation

When doing propagation, the momentum and weight decay values are chosen to reduce oscillation during stochastic gradient descent.

3.2.2 Different types of layers:

Convolutional layer
In a convolutional neural net, the parameters of each convolution kernel are trained by the backpropagation algorithm. There are many convolution kernels in each layer, and each kernel is replicated over the entire image with the same parameters. The function of the convolution operators is to extract different features of the input. The capacity of a neural net varies, depending on the number of layers. The first convolution layers will obtain the low-level features, like edges, lines and corners. The more layers the network has, the higher-level features it will get.

ReLU layer
ReLU is the abbreviation of Rectified Linear Units. This is a layer of neurons that use the non-saturating activation function . It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other functions are used to increase nonlinearity. For example the saturating hyperbolic tangent, and the sigmoid function. Compared to tanh units, the advantage of ReLU is that the neural network trains several times faster.

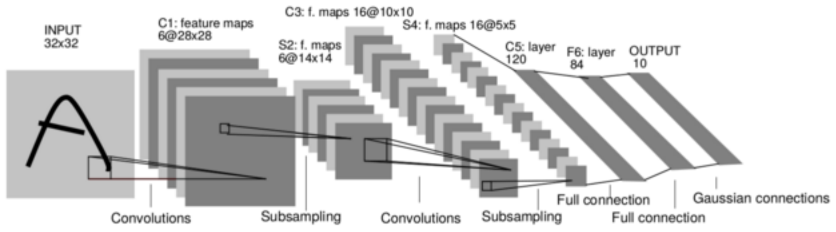
Pooling layer
In order to reduce variance, pooling layers compute the max or average value of a particular feature over a region of the image. This will ensure that the same result will be obtained, even when image features have small translations. This is an important

operation for object classification and detection.

Dropout method
Since a fully connected layer occupies most of the parameters, it is prone to overfitting. The dropout method is introduced to prevent overfitting. At each training stage, individual nodes are either ”dropped out” of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. By avoiding training all nodes on all training data, dropout decreases overfitting in neural nets. The method also significantly improves the speed of training. This makes model combination practical, even for deep neural nets.

Loss layer
It can use different loss functions for different tasks. Softmax loss, used in our model, is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in [0,1]. Euclidean loss is used for regressing to real-valued labels [-inf,inf]

Figure 5. Example of CNN architecture



3.3 Architecture

The model of a CNN can be organized practically using two major methods - graphs or sequential. For the purpose of character recognition, we use the sequential model of a LeNet 5 architecture which can be visualized as a stack of 8 layers. [4]

1. The initial layer is a convolutional layer of size 32x32 even though the input images are of size 28x28. The reason for this is, that the relevant features are then guaranteed to be contained in all feature maps and not get lost because they are near the boundary. So convolutional filters of size 32 are used with a convolution kernels of size 3.

2. The next layer is the activation layer that uses a $ReLu(RectifiedLinearUnits)$ layer that increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

3. The maxpooling layer, in charge of downsampling the spatial dimensions of the input, takes 2x2 receptive fields in order to reduce variance. Note that this discards exactly 75% of the activations in an input volume (due to downsampling by 2 in both width and height).

4. The dropout layer randomly sets a fraction 0.25 of input units to 0 at each update during training time, which helps prevent overfitting.

5. The flatten layer converts the n dimensional input to 1 dimension.

6. This is followed by another activation and dropout layer.

7. The final layer, the loss layer, uses $Softmax$ which is used for predicting a single class of K mutually exclusive classes

The model is compiled using a loss function, *categorical_crossentropy*, which is a multiclass logloss and *adadelta* as the optimization function, which is a per-dimension learning rate method for gradient descent.

3.4 Hidden Markov Model

3.4.1 Overview

A hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. Often pattern recognition problems like the one at hand, have an element of uncertainty and randomness in the form of error from the source. Probabilistic models are used to increase accuracy in such a scenario. As such, we use the Hidden Markov Model to increase upon the accuracy of the output we obtain from the Convolution Neural Network. The Hidden Markov Model treats data as a sequence of observations, while using hidden states that are connected to each other by transition probabilities.

An HMM is characterized by the following:

1. N, the number of states in the model. M, the number of distinct observation symbols.

2. A, the transition probability distribution.

3. B, the observation symbol probability distribution for each state.

4. π, the initial state state distribution.

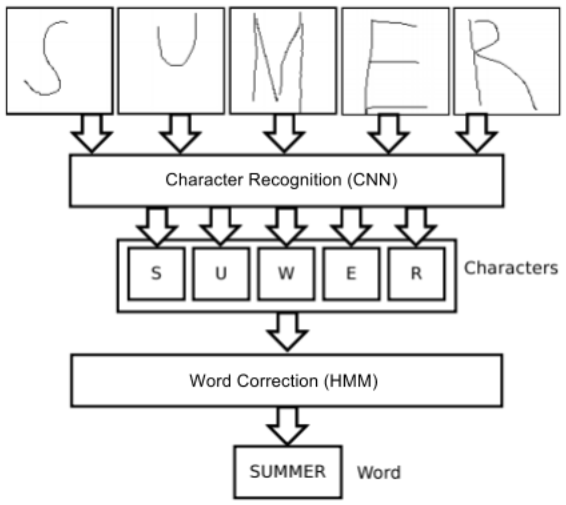
In contrast to a knowledge-based approach, HMMs use statistical algorithms that can automatically extract knowledge from samples. Hence, the performance of the model can be enhanced by adding more samples.

3.4.2 Classifiers

We utilize the HMM as a function that takes a string of characters as input and outputs a word. For this purpose, two kinds of word classifiers have been implemented. [1]

1. Forward-classifier
2. Viterbi-classifier

Figure 6. A flowchart that shows the classification process



On giving the classifier an input I, the following steps are performed to calculate the output: The probability of I is calculated for all HMMs contained in the classifier:

1. I is translated into a sequence of observation symbols O = O1 , O2 , ..., On. Here, I is a string of characters and the output of the classifier is a word. Every character in the string is simply translated to the corresponding observation symbol. There are also special observations for the start and end states. This is explained in more detail in the following section.
2. The Forward algorithm/Viterbi algorithm is then used to calculate the probability of O given the HMM.
3. The output symbol with the highest probability in the previous step is returned as output.

The following main parameters must be supplied when a classifier is created:

1. The set of possible output symbols and corresponding training examples.

2. The initialization method that should be used by the HMMs.
3. A binary variable, specifying if the training examples should be used to train the model with the Baum-Welch training algorithm.

3.4.3 Topology

The two methods used for word classification call for two kinds of topologies for the HMM.

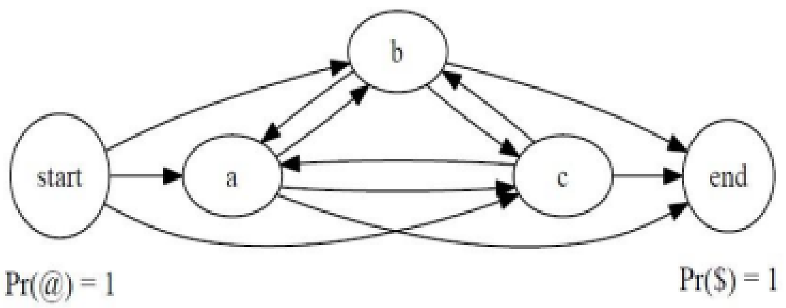
Forward classifier
For the forward-classifier, a separate model for each word in the dictionary is generated. It is natural to implement one HMM for each of the words when the vocabulary is small. When the vocabulary is larger, this approach may have performance problems. The initialization of the transition and emission matrices are done in such a way that the following properties are fulfilled:

1. The beginning state start will always emit the special symbol @ and the end state will always emit the special symbol $.
2. The beginning state start will always transition to the first normal state.
3. The ending state end always transitions back to the beginning state start.
4. All other states always transition forward to a state that has not been visited since the last visit to start.

If the word has n letters, there will be n + 2 states in the corresponding HMM. Special beginning and end states are included because multiple training observation sequences are compared against one another to give the final output word classification. The sequence is then used as input to the Baum-Welch algorithm.

Viterbi classifier
With a large vocabulary using a single HMM can be beneficial. A single model has the additional benefit that it can learn common patterns in words such as ”ing”. The second word classifier that is implemented, called Viterbi-classifier, uses a single HMM for the whole vocabulary.

Figure 7. Topology for Viterbi classifier



The single HMM for Viterbi-classifier has states that have transitions to all other states. It has 28 hidden states (26 for the 26 Latin letters and 2 with special emissions and $). The transition matrix is a 28*28 matrix, which is estimated from the lexicon analysis, using the method explain below. For example, if there are only three words in the vocabulary: DOG, CAT and CAP. Then the probability of going from A to T is 0.5, A to P is 0.5 and A to other letters is 0. According to the parameter estimation method just explained, the transition probability matrix only needs information from every two successive letters in the words. The observations are the letters observed from the character classifier or the CNN. The observation probability matrix is created from observations when testing the character classifier. For example, if we have 10 test example images for A, and 5 of them are classified to be A, 3 to be B and 2 to be C, then P(observation = A) = 0.5, P(observation = B) = 0.3 and P(observation = C) = 0.2. For this example, the row for A in the probability matrix will be set to $[0.5, 0.3, 0.2, 0, 0, ..., 0]$.

So given the HMM described above, the classification of a string of characters can be done in the following way:

1. Use the string as an observation sequence and apply the Viterbi-algorithm to get the most probable sequence of states.

2. Calculate the similarity of the resulting string to all possible output words. As similarity measure we use the hamming distance.

3. Return the most similar word.

3.5 Initial Parameter Selection

Hidden Markov Models can be efficiently trained by the Baum-Welch algorithm, which is an iterative process for estimating parameters for HMMs. As an iterative algorithm, BW starts from an initial model and estimates transition and emission probability parameters by computing expectations via the Forward-Backward algorithm. The algorithm sums over all paths containing a given event until convergence is reached. Since the Baum-Welch algorithm is a local iterative method, the resulting HMM and the number of required iterations depend heavily on the initial model. There are many ways to generate an initial model, some techniques consider the training data while others do not. The two popular initialization strategies used include, count-based and random. The random initialization strategy assigns the values in the transition and emission matrices to random values. For the count-based initialization the emission matrix is assigned based on information from the training examples. The transition matrix is then assigned so that all states get the same probability of transferring to all reachable states. For the purpose of this project, random initialization of the HMM model is done.

4 Results

1929 Training samples
275 Validation samples
300 Test samples

Epoch	Accuracy	Validation Loss	Validation Accuracy
1	0.0394	3.2288	0.0909
2	0.0918	3.0668	0.0691
3	0.2297	2.2098	0.3709
4	0.4412	1.1903	0.7273
5	0.5816	1.3283	0.6182
6	0.6941	0.6312	0.8255
7	0.7403	0.5697	0.8473
8	0.7911	1.2109	0.6545
9	0.8154	0.3936	0.8800
10	0.8652	0.5317	0.8291
11	0.8766	0.4252	0.8945
12	0.8927	0.3788	0.8909

Test accuracy: 0.923333333333

Figure 8. Segmented Characters inputted to HMM

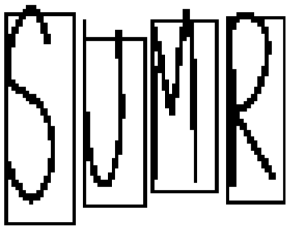
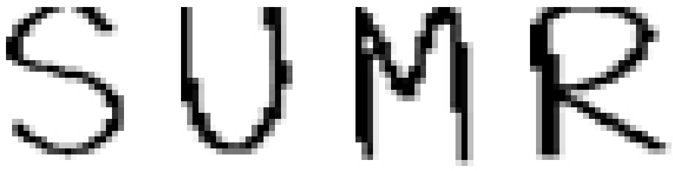


Figure 9. Individual Segmented Characters after change in size



The above input was recognized as the word 'SUMMER' by the HMM, because the given word was not in the dictionary, and SUMMER was the closest word to it.

5 Observations

We can see from the result above that the CNN after 12 epochs has a very good validation accuracy, as well as test data accuracy which was 92%. As mentioned above, the HMM used the entire word to help predict what word it should have been, given it's limited dictionary. Again, the above input was recognized as the word 'SUMMER' by the HMM, because the given word was not in the dictionary, and SUMMER was the closest word to it. It was also observed that the segmentation, and thus accuracy was highly dependent on the pre-processing.

5.1 Limitations

Even though the CNN has a very high accuracy, it is important to note that the test data was not very noisy. Hence, the CNN is not yet equipped to handle very noise inputs (such as messy handwriting, and connected words). The HMM's output could not be given over a wide range due to dataset limitations. The dataset given was purely characters, and we have made our own input images to test the HMM as can be seen in the above example. These images are not very extensive, hence not note-worthy. Also, while the dataset does not have a wider dictionary, random word inputs will not be recognized correctly. The output accuracy is highly dependent on the pre-processing and fixed parameters. Thus, estimating parameters dynamically would be more useful.

6 Conclusions

After many experiments, we find that CNN is ideal to do handwriting recognition given a proper and diverse dataset. We also realize the many limitations of our system, and how they affect the final output. We ascertain that an HMM is very useful to help predict words given some segmentation errors and even spelling errors and the combination of the CNN and HMM will possibly lead to great results with better datasets as well.

7 References

1 - Fabian Alenius, Kjell Winblad and Chongyang Sun, UPPSALA UNIVERSITY
2 - Scale space technique for word segmentation in Handwritten Document - R. Manmatha and Nitin Srimal
3 - R. Buse, Z. Q. Liu, and T. Caelli, “A structural and relational approach to handwritten word recognition.”
4 - http://yann.lecun.com/exdb/lenet/