HANDWRITTEN WORD RECOGNITION SYSTEM BY HMM MODEL

HAOXIANG YOU [YOUHAOX@SEAS.UPENN.EDU], JIA SHEN [JIASHEN@SEAS.UPENN.EDU],

ABSTRACT. In this project, we proposed different approaches of handwritten words' recognition and compared their accuracy. The segmented letters are recognized by a Convoluted Neural Network and hidden Markov model is used to improve the recognition accuracy. The HMM method with Viterbi algorithm outperformed Smoothing algorithm and Bayes filter approach with highest test accuracy.

1. INTRODUCTION

Optical Character Recognition (OCR), a technology of converting the image of handwritten texts into machine readable forms, has been a popular topic since the past 60 years because of its various applicable environments. OCR technology improves the information accessibility of users: the digital documents can be easily searched, edited, and repurposed compared to traditional handwritten documents. In particular, we focus on improving the recognition accuracy of the handwritten words that has been segmented into several letters. The solution is in open-vocabulary space, where the result is not confined in a small group of vocabularies. Our solution contains two part: the handwritten letter classifier and a state transformation model. A Convoluted Neural Network (CNN) is used to classify each separated letters and provides the baseline prediction accuracy. An additional hidden Markov model (HMM) is used to learn the correlation between adjacent letters, which is not trivial in words. In this project, we aim to validate the recognition accuracy improvement when using HMM and compare different HMM-based algorithms. Although we assume segmented letter scenarios for simplicity, the results provides important insights into the choice of HMM algorithms in OCR.

1.1. **Contributions.** (1). HMM improves the handwritten words' recognition accuracy compared with simple letter recognition. (2). HMM method with Viterbi algorithm gives the best accuracy compared with Smoothing algorithm and Bayes filter approach.

2. BACKGROUND

We build a HMM based handwritten word recognition system as shown in Figure 1. The word has already been segmented into gray picture of letters, defined as $I_i \in \mathbf{R}^{28\times 28}$. The Convoluted Neural Network outputs the probability distribution of the picture of letters, which is used as the observation $o_i \in p^{26}$ in the HMM structure. The state of HMM is the believed letters $\mathbf{X}_i \in \{0, 1, ..., 25\}$. In the HMM structure, the state transformation and observation matrix is defined as $T \in \mathbf{R}^{26\times 26}$, and $M \in \mathbf{R}^{26\times 26}$ respectively. We assume the letter recognition accuracy is α, the length of the word is n.

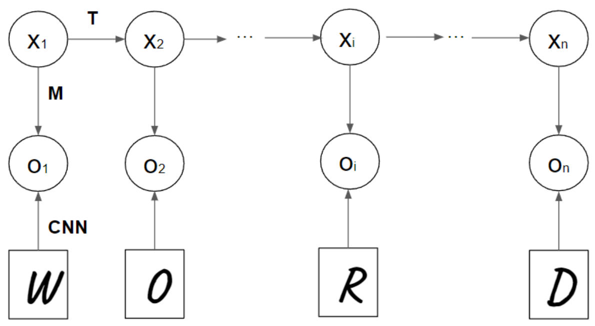


FIGURE 1. Handwritten recognition system combining HMM and CNN. CNN outputs the probability distribution of the letter as the observation of HMM.

3. Related Work

HMMs have been widely used in handwritten recognition tasks because it can model the inter-letter co-relation and learn the sequence of letters. The application of HMM fit into two categories in handwritten recognition. Some HMMs learned the comprehensive dictionary of words as the states [7], but the size of dictionary confined its applications. Other HMMs, which we used, modeled the single letters as the states [9]. Many other literature of HMM focused on the feature extraction from images [10]. *Krevat* [8] proposed a HMM with Viterbi algorithm to predict the most likely sequence of letters and compared with baseline Bayes Filter method. Despite the widely use of HMM, different implements of HMM algorithms is still unexplored. Comparing different HMM algorithms would provide significant guidance on the implement of HMM in handwritten recognition.

4. Approach

4.1. **Observer:** We assume that the word is already segmented, so the observer is the model that output the prediction of handwritten letters. The parameters of the Neural Network is defined as θ.

4.1.1. *Train CNN.* The Convoluted Neural Network[1] is commonly applied to visual imagery. We propose a CNN with structure similar to [2] to recognize the letter. The network can be expressed as a function mapping the gray image matrix to the probability distribution of letters:

$$p_i = f_\theta(I)$$

The model is trained on *A-Z Handwritten Alphabets* [3] handwritten letter dataset with $516,000$ instances. The 26 letters are not uniformly distributed but the training number is sufficient for each letter. The training parameters is set as batch size 512 and epoch number 10. The log-softmax function is used to output the probability distribution during training in order to punish the bigger errors in likelihood space [4].

4.1.2. *Baseline Prediction.* The baseline to predict the word is to find the most-likely prediction of each letter and combine those letters into a word. The baseline recognition accuracy is computed as:

$$P(word) = \alpha^n$$

Note that for long words ($n \geq 8$), even the letter recognition accuracy is high, the prediction accuracy can be low. ($0.9^8 = 0.430$).

4.2. **HMM model:** We model the handwritten recognition system as hidden Markov model. Given a word contains n letters, we define our state as random variable $\mathbf{X}_i \in \{0, 1, \ldots 25\}$, which represent the alphabet of individual letter. We define the time-step as the position of each letter appears in the word.

4.2.1. *Transition:* We defined a time-variant transition process, where we consider the position of each letter appears. Thus, for a word contains n letters, we obtained $n-1$ transition matrix. The ith Transition $T_i \in R^{26\times 26}$, where each entry t_{xy} defined the probability of ith letter being 'x' and followed by a letter 'y'. The transition matrix is obtained from counting. Given a common used word list,

$$t_{xy}^{(i)} = \frac{M}{N}$$

,where M is the number of vocabulary whose ith letter is "y" and whose $i+1$th letter is "x", N is the number of vocabulary whose ith letter is "y".

4.2.2. *Initial Distribution:* Initial distribution is obtained from counting, where we count the frequency of certain alphabet appear at the first place among all common used word.

4.2.3. *Likelihood:* likelihood reflect how truthful an observation is. In this project we establish two way to calculate the likelihood. The first way is easier. The observer can predict a probability which it believes how likely the picture is as a certain alphabet. Thus, we directly use the observer's belief as our likelihood. The second way is we establish an emission matrix $E \in R^{26\times 26}$by using data from cross validation dataset. We take the letter with highest probability predicted by observer as its prediction. Knowing the ground truth, we can obtain the emission matrix also by counting.

$$e_{xy} = \frac{M}{N}$$

,where e_{xy} is entry of emission matrix, M is the number of our observer predict "x" where the ground truth is "y". N is the number of state being "y".

4.3. **Algorithms:** We use three different algorithms:Bayes filter, Smooth and Viterbi's algorithm to solve this HMM model.

4.3.1. *Bayes filter:* Given observations up to time k, compute the distribution of state at time k. The prediction can be updated recursively:

$$P(X_k|Y_1,\dots,Y_k) = \frac{P(Y_k|Y_1,\dots,Y_{k-1},X_k)P(X_k|Y_1,\dots,Y_{k-1})}{P(Y_1,\dots,Y_{k-1})} \tag{1}$$

$$= \frac{P(Y_k|Y_1,\dots,Y_{k-1},X_k)\sum_{X_{k-1}} P(X_k|Y_1,\dots,Y_{k-1},X_{k-1})P(X_{k-1}|Y_1,\dots,Y_{k-1})}{P(Y_1,\dots,Y_{k-1})} \tag{2}$$

$$= \eta P(Y_k|X_k)\sum_{X_k-1} P(X_k|X_{k-1})P(X_{k-1}|Y_1,\dots,Y_{k-1}) \tag{3}$$

where η is a normalization constant, $P(Y_k|X_k)$ is the likelihood, $P(X_k|X_{k-1})$ is the transition and $P(X_{k-1}|Y_1,\dots,Y_{k-1})$ is the recursive term, representing the previous prediction.

4.3.2. *Smooth:* Compared to Bayes filter, Smooth algorithm not only uses the observations up to current time-step but also use the future observation to make a prediction. That is $P(X_k|Y_1,\dots,Y_n)$. To efficiently compute $P(X_k|Y_1,\dots,Y_n)$, we introduced inter-medium variables: $\alpha_k(x)$ and $\beta_k(x)$, which can be updated recursively.

$\alpha_k(x)$ is defined as $P(Y_1,\dots,Y_k,X_k=x)$ and can be recursively updated by:

$$\alpha_1(x) = \pi_x M_{x,y_1} \tag{4}$$

$$\alpha_{k+1}(x) = M_{xy_{k+1}}\sum_{x'}\alpha_k(x')T_{x'x} \tag{5}$$

where is M is the likelihood emission matrix, T is the transition matrix, π is the initial distribution.

$\beta_k(x)$ is defined as $P(Y_{k+1},Y_{k+2},\dots,Y_n|X_k=x)$ and can be updated by:

$$\beta_n(x) = 1 \tag{6}$$

$$\beta_k(x) = \sum_{x'}\beta_{k+1}(x')T_{xx'}M_{x'y_{k+1}} \tag{7}$$

after obtained $\beta_k(x)$ and $\alpha_k(x)$, we can obtained

$$P(X_k|Y_1,\dots,Y_n) = \eta\alpha_k(x)\beta_k(x) \tag{8}$$

where η is a normalized constant

4.3.3. *Viterbi:* Compared to smooth algorithm, which predict the maximum likely state at single time-step given all the observations, Viterbi algorithm predict the jointly maximum likely states for all time-step given all observations. where

Algorithm 1 Viterbi

```
Initialize:
    $\delta_1(x) = \pi M_{xy_1}$
    $\text{parent}_1(x) = \text{null}$
Updates:
    $\delta_{k+1}(x) = \underset{x'}{max}\ \delta_k(x')T_{x'x}M_{xy_{k+1}}$
    $\text{parent}_{k+1}(x) = \underset{x'}{argmax}\ (\delta_k(x')T_{x'x})$
Predict:
    $X_n = \underset{x'}{argmax}\ \delta_k(x')$
    $X_k = \text{parent}_{k+1}(X_{k+1})$
```

the M is the emission matrix and T is the transition matrix, same as any other algorithmns described above.

5. Experimental Results

5.1. **Observer.** The training loss shown in Figure 2 converges after 5 epoches.

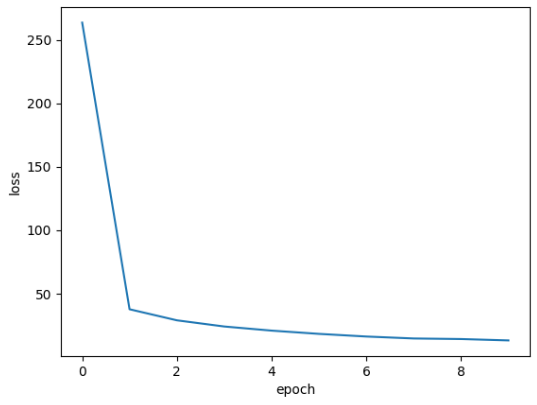
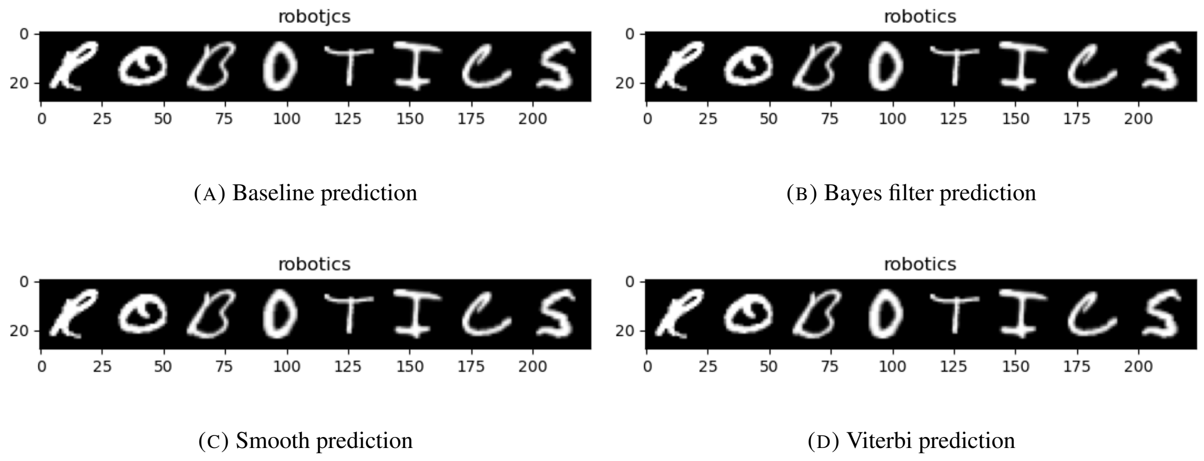


FIGURE 2. The training loss of handwritten recognition CNN

The test accuracy of the CNN is 97.8%.

5.2. **HMM.** The figure 3 gives an example of the prediction of a word ”robotics”, where the image is how the handwritten word looks like and the title is prediction make by our model. The baseline prediction have difficulty to distinguish 6th letter between ”I” and ”J”, however, by introducing HMM model all three different algorithm can correctly predict the word ”robotics”.



(A) Baseline prediction (B) Bayes filter prediction

(C) Smooth prediction (D) Viterbi prediction

FIGURE 3. Prediction of ”Robotics” using likelihood directly output by observer

The following tables give the prediction accuracy of different algorithms. Overall, combined with HMM model, the accuracy of prediction increases. The likelihood obtained directly by observer perform better than the likelihood obtained use a pre-determined emission matrix. The Viterbi algorithm with the likelihood obtained from observer gives the best performance with the average accuracy of prediction 91%, which is 9% higher than use observer only.

6. Discussion

In the previous section, we have seen that the way to calculate the likelihood of single observation can affect the performance of our system. Use the probability output by neural network directly out performance than the likelihood

Likelihood by emission matrix %				
Word or Average	Baseline	Bayes Filter	Smooth	Viterbi
'Learning'	80.0	84.0	84.0	85.0
'In'	90.0	90.0	93.0	95.0
'Robotics'	82.0	88.0	90.0	88.0
'University'	73.0	76.0	84.0	82.0
'Of'	92.0	92.0	92.0	92.0
'Pennsylvania'	75.0	78.0	81.0	78.0
Average	82.0	84.7	87.3	86.7

Likelihood by observer directly %				
Word or Average	Baseline	Bayes Filter	Smooth	Viterbi
'Learning'	80.0	86.0	87.0	89.0
'In'	90.0	93.0	96.0	98.0
'Robotics'	82.0	88.0	89.0	89.0
'University'	73.0	88.0	88.0	92.0
'Of'	92.0	93.0	95.0	94.0
'Pennsylvania'	75.0	78.0	79.0	84.0
Average	82.0	87.7	89.0	91.0

obtained by argmax and look up the emission matrix. We think the reason for the lower performance of taking argmax is that we throw out information about a specific handwritten. For example if someone write a A really similar to R, the observer may catch this feature by assign a highest probability for 'R' but also keep a relevant high probability for 'A'. By taking argmax, the good catch-up is throw away while we should keep the likelihood for underlining letter being 'A' high. We can probably construct a new way to obtain likelihood that contain as much as information that observer can catch-up. For example, we can treat the whole distribution output by observer as an observation rather rather simply take argmax and calculate likelihood assume the observation is from certain distribution.

Another issue for our current model is the transition may be a little bit too rough. Vocabulary like 'Pennsylvania' is very identical but our model still not predict it well. We probably can obtained a transition matrix characterize both position and total length of the word to improve the performance.

References

[1] Valueva, M.V., Nagornov, N.N., Lyakhov, P.A., Valuev, G.V. and Chervyakov, N.I., 2020. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation, 177, pp.232-243.
[2] Alwzwazy, H.A., Albehadili, H.M., Alwan, Y.S. and Islam, N.E., 2016. Handwritten digit recognition using convolutional neural networks. International Journal of Innovative Research in Computer and Communication Engineering, 4(2), pp.1101-1106.
[3] Pratik Chaudhuri, ESE 650 2021 Class Notes: https://pratikac.github.io/pub/21_ese650.pdf
[4] https://discuss.pytorch.org/t/logsoftmax-vs-softmax/21386
[5] https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format
[6] https://www.kaggle.com/rtatman/english-word-frequency
[7] R. Nag, K.H. Wong, and F. Fallside, 1988. Script recognition using hidden markov models. In Proceedings of ICASSP.
[8] Krevat, E. and Cuzzillo, E., 2006. Improving off-line handwritten character recognition with hidden markov models. Transaction on Pattern Analysis and Machine Learning, 33.
[9] S.N. Srihari, I.J. Hull, and R. Chowdhury, 1983. Integrating diverse knowledge sources in text recognition. In ACM Trans. on Office Automation Systems.
[10] H. Bunke, M. Roth, and E.G.Schukat-Talamazzini, 1995. Off-line cursive handwriting recognition using hidden markov models. Technical Report IAM-94-008, University of Bern.