Assisting Fuzzy Offline Handwriting Recognition Using Recurrent Belief Propagation

Yilan Li, Zhe Li, Qinru Qiu
Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244, USA
Email: {yli41, zli89, qiqiu}@syr.edu

***Abstract*—Recognizing handwritten texts is a challenging task due to many different writing styles and lack of clear boundary between adjacent characters. This problem has been tackled by many previous researchers using techniques such as deep learning networks and hidden Markov Models (HMM), etc. In this work we aim at offline fuzzy recognition of handwritten texts. A probabilistic inference network that performs recurrent belief propagation is developed to post process the recognition results of deep convolutional neural network (CNN) (e.g. LeNet) and form individual characters to words. The post processing has the capability of correcting deletion, insertion and replacement errors in a noisy input. The output of the inference network is a set of words with their probability of being the correct one. To limit the size of candidate words, a series of improvements have been made to the probabilistic inference network, including using a post Gaussian Mixture Estimation model to prune insignificant words. The experiments show that this model gives a competitively average accuracy of 85.5%, and the improvements provides 46.57% reduction of invalid candidate words.**

I. Introduction

In recent years, many studies focus on recognizing handwritten words. Handwritten words demonstrate high variabilities because each person possesses his/her own unique writing style. Furthermore, clear boundaries cannot always be found between characters in handwritten text. Adjacent characters sometimes are connected or overlapped. This significantly increases difficulty for detection algorithms solely based on pattern matching. Without clear boundary, two characters that are close to each other may be recognized as one character or one character may be split into two characters. These errors are usually referred as *insertion* or *deletion errors*. Recognitions of handwritten characters in offline situation are more challenging, because it does not have dynamic representations of hand writing trajectories, which is a useful feature for classification.

Some of recent approaches apply Histograms of Oriented Gradients (HoGs) [1], discrete HMM [2,3] or deep neural networks [4,5] to recognize handwritten words. In general, these works investigate two directions to improve the recognition accuracy: 1) Searching for the set of more robust features that are orientation, distortion insensitive; 2) Incorporate language and dictionary information with the pattern recognition. S. Yao [6] used a method based on sequences of HoG feature vectors. This method normalizes and divides the input image into equal-sized cells, and then organizes HoG descriptors into horizontal and vertical directional features vectors. Discrete HMM has been successfully used for handwritten Arabic word recognition by M. Dehghan *et al.* [7]. They use the histogram of 4-directional chain code as feature vectors, by using a moving window scanning the input image from left to right. However, directional features of handwritten words are variable and they are hard to recognize with rotation and distortion. A. Gupta [8] improved this by using Fourier descriptors. However, exactly segmenting words into individual characters is essential, which is less likely to achieve when the input is noisy. Y. LeCun [9,10] proposed an efficient multilayer convolutional neural network (CNN) for recognizing both handwritten digits and characters.

To integrate dictionary information with character recognition, [9] applies standard grammar graph to select the output from the CNN based character recognizer and form the selected characters into words. Although effective, only if both recognition graph and standard grammar graph reach the end nodes will an acceptable sequence of input symbols be selected and the standard grammar graph is not recurrent, therefore, it is not capable of correcting the deletion errors, which is very likely to occur when no well-defined character boundaries are found. This is improved by the multidimensional recurrent neural network (RNN) proposed by Alex Graves [11]. Trained with the image of whole words, the RNN is able to recall the spatial pattern of adjacent characters, which improves recognition accuracy. Esam Qaralleh *et al.* [12] tuned the recurrent neural network to a deep neural network with three hidden layers and two subsampling layers. Their approach segments the input word into sub-words first and then recognized sub-words using RNN. In this way, the complexity is greatly reduced. The RNN provides a comprehensive solution for spatial temporal pattern recall, however, its training is quite expensive. In [13–15], a layered framework is developed that utilizes cogent confabulation model in the upper layer to form correct words and sentences based on the characters detected by the bottom layer using pattern matching. However, similar to [9], the confabulation model assumes that the images and characters in the real word has one to one correspondence, it cannot correct the insertion and deletion error.

In this work, we aim at incorporating dictionary information to assist recognition of handwritten text images. We generalize the definition of handwritten text to any text image with irregular fonts and possible overlapped characters. To avoid expensive training of an RNN, we adopted similar layered approach as [9] and [13]. The bottom layer is a CNN for pattern recognition and the upper layer is a recurrent belief propagation network that searches for the possible words that can be formed using the detected characters. The belief

propagation network generates fuzzy outputs. The output is a set of possible words recognized from the given image and their scores. The fuzzy output can further be refined if sentence level information is provided.

The belief propagation network is a neural network constructed based on a given dictionary and the construction complexity is linear to the size of the dictionary. Unlike the models in [8] and [13], our network does not rely on accurate separation of the characters. Each neuron maintains a memory. The state of a neuron is not only determined by its input, but also its historical value. These improvements not only enable the model to correct insertion and deletion errors, but also replacement errors. If a wrong spelled word that is not in the dictionary, our system will give several most likely word candidates with the correct spelling. Some examples are given in Section III-D.

The goal of our research is to reduce the size of the output set of word candidates and to increase the ranking of the correct word within the output set. This is achieved by a carefully designed inference network and certain pruning techniques such as post Gaussian Mixture Estimation. Compared to the standard grammar graph based word detection, we have more than 5% improvement in word accuracy. Compared to the result without post Gaussian estimation, 46.57% unrelated word candidates are pruned with additional 7.2% ranking increase.

The rest of paper is organized as the follows. Section II provides the details about system architecture and algorithm. Experimental results are given in Section III and Section IV summarizes the current work and discusses future work of our research.

II. System Architecture and Algorithm

In this section, we will give the overview of the whole framework and then describe each layer in detail.

A. Network Overview

The overall framework has three layers: (1) Segmentation layer using Local Peak Finder Algorithm [16]; (2) Recognition layer using LeNet-Structured CNN; and (3) Inference layer using recurrent belief propagation network. Fig. 1 shows the flowchart of the overall framework. Its input is handwriting word images. The output is a fuzzy recognition. It consists of a set of possible word candidates and their ranking and scores.

Using Local-peak finder algorithm, we divide a word image into a sequence of segments. Then the LeNet-based recognizer gives possible class labels with probabilities. The segmentation step is a best effort approach to separate characters. Since there is not always clear boundary between characters in handwritten texts, the separation is not perfect. It is possible that a segment consists of multiple characters. Those segments

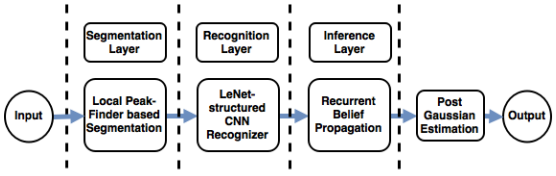


Fig. 1. Overall Framework

are detected and processed using Space Displacement Neural Network (SDNN) [17]. The details will be introduced in Section II-B.

Lastly, the belief propagation network recalls words with the highest likelihood based on the predictions from the CNN and report their ranking and scores.

B. Segmentation based on Local Peak Finder Algorithm

Space Displacement Neural Network (SDNN) has been proposed in [17] to apply CNN on text images with connected digits. The neural network is used to process each sub-image selected by a series of sliding windows, and the output is connected using a Viterbi alignment module. Naively applying SDNN to the original word image will unnecessarily increase the computation complexity. In the first step, we segment the image into separate characters or character groups and then apply SDNN on each segment. In this work, we improved the local peak finder algorithm in [16] to search for segmentation point. The algorithm is shown in Algorithm 1.

Algorithm 1: Local Peak Finder Algorithm: pseudo code

```
True pixel peak set T_p = ∅;
Obtain column pixel value summation vector C;
Record positions and values of every valley V_i and peak P_t in
  C;
Calculate maximum different Δ between peak and valley ;
if column pixel value < 0.6Δ then
    Remove the column from consideration;
end
Initialize found indicator found = true;
while position V_i in valley set not the end do
    if found = true then
        Find peak position P_t prior previous to V_i;
        Set found = false;
    end
    Mark position V_imin of min valley between P_t and V_i;
    if V_imin ≠ V_i then
        Set frame between V_imin between V_i as decision
          region D;
        Mark position P of max peak in decision region D;
        if pixel value of P ≥ threshold T then
            Add P to true peak set T_p;
            Set found = true;
        end
    end
    Move V_i to next valley position;
end
Separate input image at position value in set T_p;
```

In this work we consider black-white images with white background and black foreground. The white pixel value is 255 and black pixel value is 0. Firstly, we calculate the summation of the pixel values for each column and record the positions and values of peak (column with local maximum pixel value) and valley (column with local minimum pixel value). The maximum difference between the peak and valley are calculated and denoted as Δ. If a columns pixel value is less than 0.6Δ, it will be removed from consideration. Therefore, only those columns with large number of white pixels will be considered as potential point for segmentation. Starting from the first valley position V_i, we look for the first peak P_t to the left of V_i and the region between these two is set as decision region. When the pixel values in this region

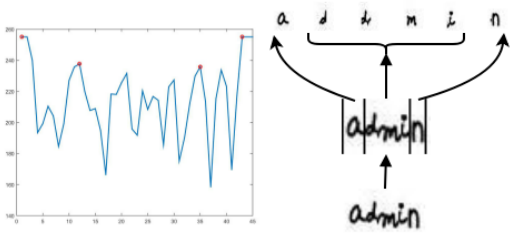


Fig. 2. Example for Segmentation Layer

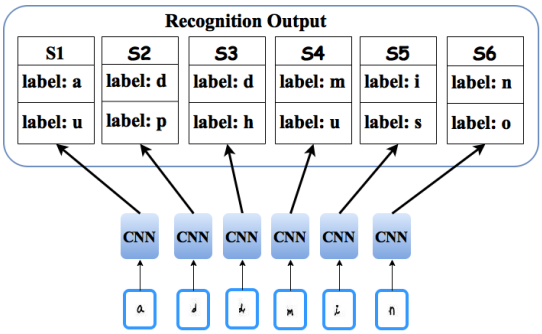


Fig. 3. Example for Recognition Layer

are all greater than the value of V_i, V_i is move to the next valley position. If there exists a point V_{imin} with smaller pixel value than V_i and the peak between $[V_{imin}, V_i]$ is greater than a particular threshold T, we will segment the image at the location of the peak, and continue processing the remaining image using the same algorithm.

The segmentation algorithm is only a best effort approach. It is possible that multiple connected characters will be included in the same segment. Those segments whose width is less than the average character width T_w will be processed directly by the CNN for pattern matching. T_w is obtained from the training set. For segments whose width is greater than T_w, a moving window that is T_w wide is used to scan through the segment from left to right with one pixel step size. Each image selected by the window will be processed by the CNN.

Fig. 2 shows an example of the segmentation step. The input word image is shown at the bottom. The column pixel value is plotted and three segmentation point are identified and highlighted. The middle segment is wider than average. Applying a moving window to this segment, we got four images for this area. Overall six images will be sent to CNN for pattern recognition.

C. LeNet-Structured CNN Recognizer

We use the CNN structure defined by Berkeley Vision and Learning Center [18] for pattern recognition. It is trained to recognize 26 English alphabets. The structure is the same as LeNet. The input is a 28×28 image and output is a set of possible characters and their probabilities. Please note that the image generated from the segmentation layer has with T_w, which is less than 28. They are padded with white space to make the size 28×28. The base learning rate of the training is set to 0.001 and number of iterations is set to 5000.

Each recognizer predicts a vector of N most likely labels for every segment. Therefore the input of belief network in next

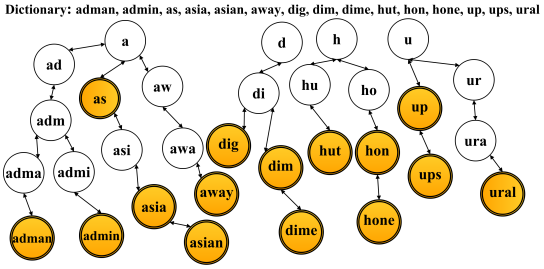


Fig. 4. Example for Neuron Pool

layer is a sequence of N dimensional vectors. Each vector represents a set of possible candidates perceived at specific location in the input image.

Using the segmentation results from Fig. 2 as an example, we show how the recognition layer works in Fig. 3. Here N is set to 2.

D. Recurrent Belief Propagation Network

The inference layer of belief propagation is a neural network that consists of three types of neurons: input neurons ($\mathcal{I}$), interpretation neurons ($\mathcal{P}$), and dictionary neurons ($\mathcal{D}$).

Every substring of characters in a dictionary word starting from the first character corresponds to a dictionary neuron. We denote a dictionary neuron as $\mathcal{D}_\alpha$, where α is the substring associated to the neuron. For example, the word "admin" is associated to 5 dictionary neurons, $\mathcal{D}_a$,$\mathcal{D}_{ad}$, $\mathcal{D}_{adm}$, $\mathcal{D}_{admi}$, and $\mathcal{D}_{admin}$. We can further divide the dictionary neurons into two categories: neurons that represent real dictionary words or neurons that represent substring of real words.

All dictionary neurons are connected in a Trie [19] structure, and all connections are bi-directional. That means, neurons i and j are connected, if i is the prefix of j or vice versa. Furthermore, if i is the prefix of j, then we call the link from i to j as *prediction link*, and the link from j to i as *feedback link*. For example, there is a prediction link from neuron $\mathcal{D}_a$ to neuron $\mathcal{D}_{ad}$, and a feedback link from $\mathcal{D}_{ad}$ to $\mathcal{D}_a$. However, there is no connection between $\mathcal{D}_a$ and $\mathcal{D}_{adm}$ or any other neurons in previous example. The bidirectional connections form a recurrent network.

In this work, the dictionary neurons are generated using Mieliestronks list [20]. This dictionary has 58027 English words with average length of 8 characters. It generates approximate 470000 dictionary neurons. We refer to the overall dictionary neurons as the neuron pool. Fig. 4 shows all dictionary neurons and their connections generated from a small dictionary with only 15 words. All neurons that correspond to real word are highlighted in orange. As we can see, the network has a tree structure.

As shown in Fig. 5, each directional link between two neurons is associated with a weight, which is set to be the log conditional probability $\log[p(s|t)/p_0]$ between the source and target neurons of the link. The probability for prediction links is collected statistically from the dictionary. For feedback links, this value is always 1.

There are 26 input neurons, each represents a possible character candidate detected by the CNN recognizer. An input neuron is denoted as $\mathcal{I}_\beta$, where β is one of the 26 English alphabet. Considering the possible errors in recognition, we add a set of 26 interpretation neurons, denoted as $\mathcal{P}_\gamma$, where

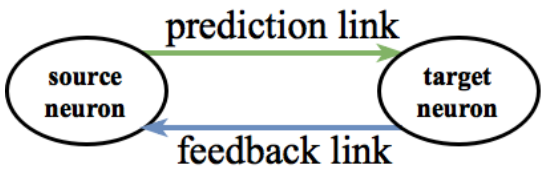


Fig. 5. Knowledge Link

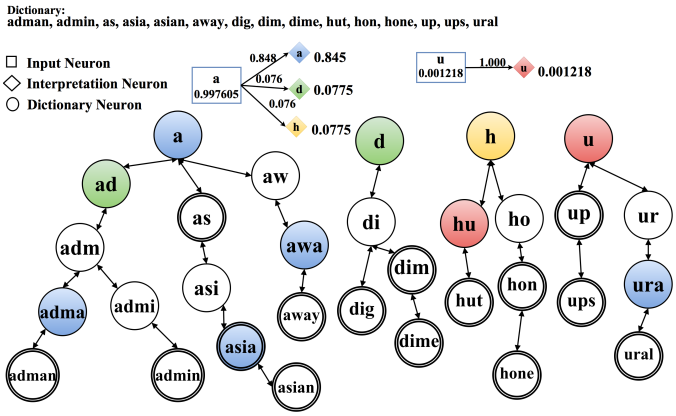


Fig. 6. Example for Inference Layer

each γ is an English character. Links are established from $\mathcal{I}_\beta$ to $\mathcal{P}_\gamma$, the weight of the link is the probability that γ is recognized as β. This information is collected from the training process. There is also knowledge links from the interpretation neuron to the dictionary neuron. If γ is the last character in the substring α, then there is a link from interpretation neuron$\mathcal{P}_\gamma$ to dictionary neuron $\mathcal{D}_\alpha$.

An example of all 3 types of neurons and their connections is shown in Fig. 6 for the 15-words dictionary. The interpretation and dictionary neurons are represented by rhombus and circles respectively. To make the figure readable, we do not show the connections between interpretation neurons and dictionary neurons, but they are reflected by neuron colors, i.e. there is a link between interpretation neuron and dictionary neuron with the same color. In this example, there is a 0.076 probability that a letter "d" will be recognized as "a", therefore, the link from $\mathcal{I}_a$ to $\mathcal{P}_d$ has weight 0.076. The interpretation neuron $\mathcal{P}_d$ will excite all dictionary neurons that end with letter "d", therefore, it has a connection to two dictionary neurons, $\mathcal{D}_d$ and $\mathcal{D}_{ad}$.

Each neuron maintains its excitation level. The excitation level of each input neuron is directly set to the corresponding class probability reported by the CNN. The excitation levels of all the other neurons are calculated as:

$$e(t) = \sum_{k \in F_l} \sum_{s \in S_k} I(s)[\ln(\frac{p(t|s)}{p_0}) - \ln(\frac{p(s|t)}{p_0}) + B] + (1-\alpha)I(t), t \in S_l \quad (1)$$

where $I(s) = e(s)/\sum_{x\in\mathcal{P}} e(x)$ if $s \in \mathcal{P}$, and $I(s) = e(s)/\sum_{x\in\mathcal{D}} e(x)$ if $s \in \mathcal{D}$, α is the diminishment ratio. Variable $I(s)$ is the normalized excitation level of neuron s, and is referred as *activation level*. The normalization is carried out across all neurons of the same type. As we can see from the equation, the excitation level of a neuron t is determined by both the input excitation and the neurons current activation. In other words, a neuron has memory. Even if it is not being excited externally, it will still remain

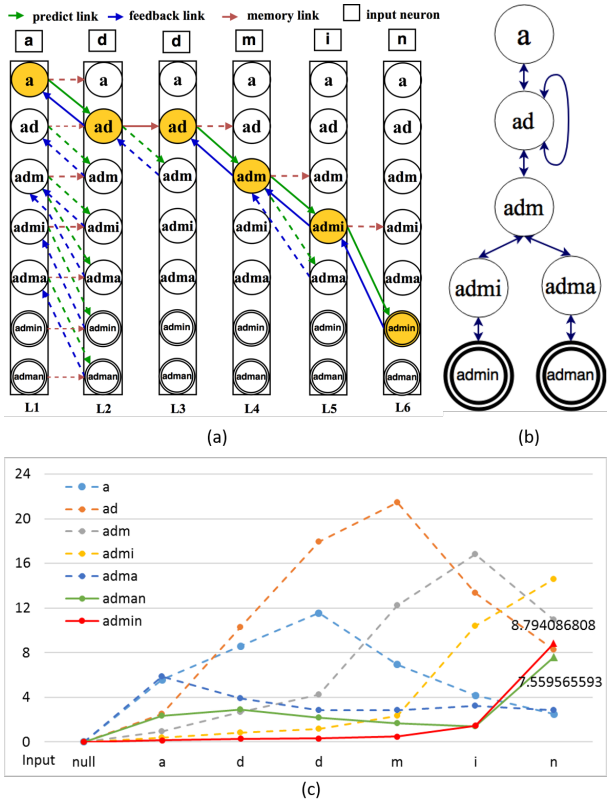


Fig. 7. (a) Confabulation Network, (b) Recurrent Belief Propagation Network, (c) Example for Neuron Excitation Level Evaluation

active. However, its activation level will diminish a specific percentage α if it does not receive input excitation. During the normalization procedure, the activation of this neuron will be inhibited as other neurons that have been excited externally keeps on increasing their excitation level. For a dictionary neuron $\mathcal{D}_\alpha$, its input comes the prediction link, the feedback link and the link from the interpretation neuron. The predictive signal predicts the next character that may be perceived, the feedback signal confirms the previous perception based on current inputs and the interpretation signal simply represents the sensory input from the recognizers. The proposed model is to certain degree similar to the HTM model [21]. However, the HTM model uses a one-bit flag to indicate prediction status, while the prediction in our model is lumped in the neuron excitation level.

During recall, the excitation level of all neurons will be updated each time a new input is received from the pattern matching layer. Please note that the normalization is performed after each update, therefore, the neurons with higher excitation will suppress those with lower excitation in a soft winner-takes-all manner. At the end of recall, a sequence of neurons will be highly activated, which form a path (or multiple paths) that lead to the predicted word(s).

Please note that if we unroll the recurrent network over time, it is actually similar to a confabulation network [22–24]. Using the network for word "admin" as an example. The belief network is shown in Fig. 7 (b). It consists of 7 dictionary neurons. Assume six images are separated (either by segmentation or sliding window) for pattern recognition as shown in Fig. 2, then the belief network will be evaluated 6

times. If we unroll the recurrent network over time and create a copy of all dictionary neurons for each evaluation interval, then we obtain a confabulation network with 6 lexicons as shown in Fig. 7 (a), each lexicon has 7 symbols corresponding to all dictionary neurons. Each symbol only connects to symbols in its adjacent lexicons and there is no connection between symbols in the same lexicon. The connections corresponding to predictive and feedback links are illustrated in Green and Blue. There is also a Red connection that links the same neuron in adjacent lexicons. This models the memory of the neurons, i.e. the neurons previous excitation level affects its current excitation level. To make the figure simple, we only show links between Lexicon 1 and Lexicon 2, however, these links should repeat between all other adjacent lexicons. The path shown by solid arrows leads to the correct word.

Assume that the recognizer recalls only 1 matching pattern for each image, i.e. the output from the recognition layer is a sequence of six 1-D vectors, and the output of the recognition layer is the sequence "a, d, d, m, i, n". Also assume that the same set of characters are triggered in the interpretation layer. These input signal will dynamically change the excitation level of neurons. Fig. 7 (c) plots how the neuron activation level changes over the time. For example, neurons "a" and "adma" first get excited when the first input "a" is received. Neurons "ad" and "adman" are being predicted then, and they further predict downstream neurons. In the next, the input "d" is received, and neuron "ad" is further excited. It sends feedback signal to neuron "a" to confirm the previous observation and continue predicting neuron "adm". The excitation level of "adma" diminishes gradually even though it got excited at the beginning, because there is no feedback or input to confirm the observation (or prediction). At the end, the word "admin" accumulates the highest excitation. Also the set of neurons "a", "ad", "adm", "admi", and "admin" has the highest activation and they identify a path that leads to the correct word.

E. Further Improvements

Sometime, certain common combination of characters or high-frequency words will reach high excitation simply from prediction, even though some of their characters are not reported by the recognizer. The last step of our work is to lower the ranking of these words or eliminate them from the candidate list. This is achieved by adding pre-processing and post-processing.

We denote estimation of the word length and start-end position constraint as pre-processing constraints. We firstly estimate the probable number N_c of characters in the word, which equals the quotient of input image pixel width divided by average character width T_w. If the word candidate has more than N_c characters, it will be eliminated. Furthermore, start-end position constraint is used to strengthened the excitation process of neurons. In the first evaluation interval, excited neurons containing only one character will get more excitation than others. Similarly, excited neurons representing a real word will get more excitation if the input from the pattern matching layer is the last. For example, adding start-end position constraint will help to differentiate the first excitation levels of "a" and "adma" in Fig. 7 (c), as "a" will get more excitation increase than "adma".

The post-processing is achieved by considering the location distribution of each character. For any English character x, we consider its location in a word is a random variable. The probability that x will be the lth letter in the word is denoted as $pr_x(l)$. We assume that this distribution follows a Gaussian Mixture Model (GMM), and the information reported by the recognizer is a sample of the distribution, based on which the whole distribution is constructed [25].

Using the GMM, a variable $Prob(w) = \prod_{i=1}^{N} pr_{x_i}(i)$ is calculated for each word w, where x_i is the ith character in w. This variable indicates the possibility that w is the correct word by considering where each character in w is located. The excitation level of word w is then adjusted based on the probability by calculating: $el'(w) = el(w) + \ln \frac{prob(w)}{p_0}$. Please note that the excitation level $el(w)$ is actually the log probability of w estimated using the inference network, therefore, the adjustment is actually calculating the product of the two probabilities to combine the prediction results from different approaches. Our experimental results show that combining with GMM will improve the ranking of the correct word by 7.2%.

III. Experimental Results

In this section, we demonstrate the performance of the handwriting recognition framework. Several simple improvement techniques are also discussed.

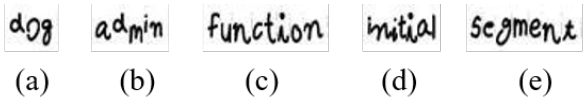
A. Experiment setup

The dataset that we use for training and testing are generated based on images from Chars74k [26]. Our dictionary is the Mieliestronks list [20]. In this list, the British spelling was preferred and American versions are deleted. Only lower words are tested in our experiments. The word images are generated using the method as mentioned in [9]. We first randomly select a word from the Mieliestronks list. For every character in the word, we then randomly select a character image from different writing styles in the Chars74k dataset and put them close together. We allow adjacent characters to connect to each other. We keep the height of the word image to be 28 pixels by scaling without changing its aspect ratio. Some sample input word images are shown in Fig. 8.

B. Training of CNN-based Recognizer Component

We train the CNN-based recognizer with a subset of Chars74K [26]. Chars74K contains 26 classes and 55 samples for each class. All the 3410 hand drawn characters in the dataset are lower-case English characters, from a to z. We use 32 samples per class for training and the rest 13 samples per class for testing. We choose 28×28 as the size of input images, which is the same as defined in [18] LeNet caffemodel.

During testing, the CNN report a set of class labels and their matching probabilities. The test accuracy of various training learning rates and iterations is reported in Fig. 9. "i_{th}" means



(a) (b) (c) (d) (e)

Fig. 8. (a) dog (b) admin (3) function (4) initial (5) segment

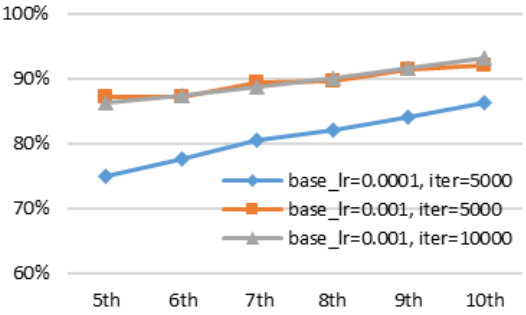


Fig. 9. Test Accuracy of CNN-based Recognizer

that the correct labels score is within the i_{th} highest predicted score. According to these results, we choose 0.001 as base learning rate and 5000 iterations to train the CNN model. To limit the complexity of probabilistic inference, we only send six highest possible class labels to the next layer.

C. Word Recognition Accuracy

We test the whole system under an environment using NVIDIA® GeForce GTX 750 with 512 CUDA cores. In the experiments, 187200 input images containing 6240 different words which are randomly selected in [20] are generated. Word candidates are selected from the highest excitation level to the lowest. We report the results from three aspects: (1) the chances that the correct word is within top 5, 10 and 20 predictions; (2) the number of word candidates; (3) expected ranking of the correct word.

Firstly, Fig. 10 shows the chances that the correct word is within top 5, top 10 and top 20 predictions respectively after adding improvement techniques. As we can see from the figure, we got 46.07% average accuracy increase if the correct word is predicted within Top 5. 36.84% accuracy improvement is got when the word is correctly predicted within Top 10. The accuracy will increase 18.75% if the correct word is predicted within Top 20.

As the end of belief propagation, the dictionary neurons corresponding to real words and whose excitation level is non-zero are reported as candidates. Large amount of word candidates means high complexity and ambiguity. Therefore, we show how the pre- and post-processing techniques can help improve the validity of our network.

Fig. 11 shows the decrease of average number of word candidates after applying different improvement techniques. Adding preprocessing constraints can help prune 58.06% irrelevant word candidates. Applying Local Peak Finder based

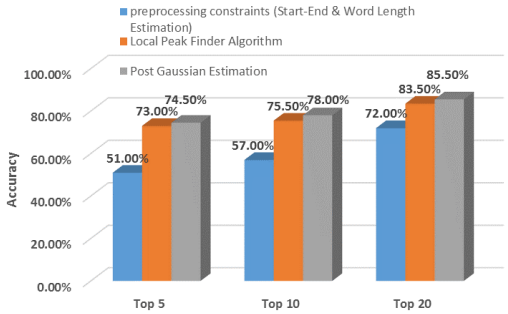


Fig. 10. Accuracy Improvement

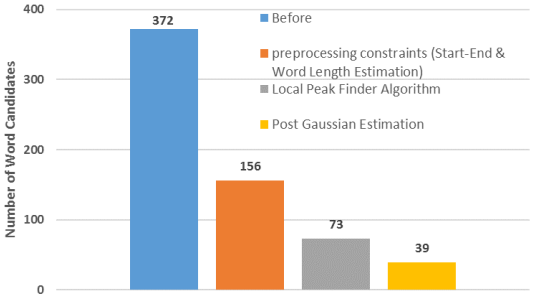


Fig. 11. Average Number of Word Candidates Improvement

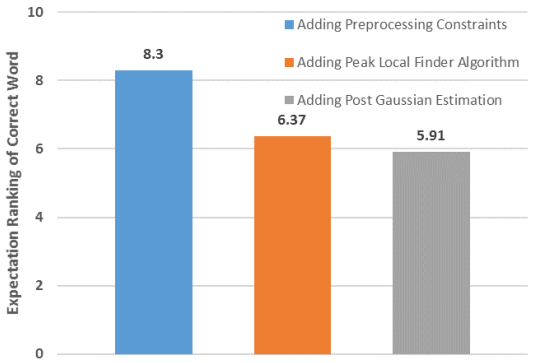


Fig. 12. Expectation Ranking Improvement of Correct Word

segmentation will further reduce 53.21% word candidates. Combining these with post Gaussian estimation gives another 46.57% reduction. Overall, the number of word candidates can be reduced to 39 on average. Fig. 12 shows the expectation ranking improvement of the correct word if it is reported within Top 20 predictions. The lower ranking means better recognition quality. The figure indicates that the expected ranking of the correct word is 8.3 when only preprocessing constraints are applied. With segmentation, this value drops to 6.37. Finally, after using post Gaussian estimation, the expected ranking of correct word reduces to 5.91.

As a base line reference, we also implement the standard Grammar Graph, which is a type of finite-state transducers [27], mentioned in [9], and use it to replace the belief network. Fig. 13 compares the accuracy of the two approaches after applying preprocessing constraints, local peak finder algorithm and post-processing. Again, the accuracy is measured by the chances that the correct word is within the Top 5, 10 and 20 predictions. The results show that our approach is 5∼8% better than the standard Grammar Graph.

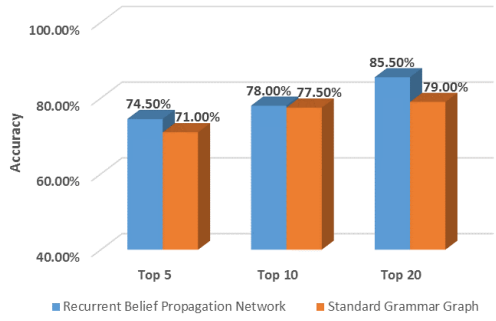


Fig. 13. Comparison for Accuracy

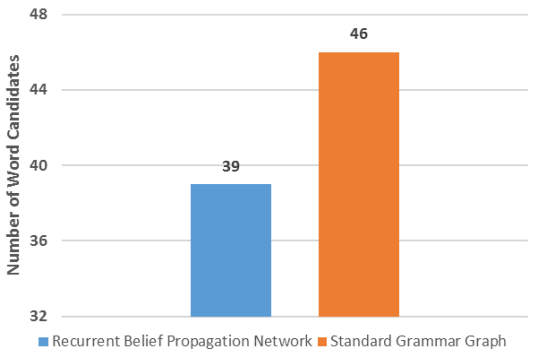


Fig. 14. Comparison for Average Number of Word Candidates

TABLE I
WORD CANDIDATES OF SAMPLE WORDS

Input Image	Output Word Candidates	Excitation Level
best	**best**	**10.350**
	test	10.1219
	beet	9.9260
	stet	8.5189
admin	**admin**	**7.8904**
	adman	5.1902
initial	**initial**	**5.4023**
	fantail	-1.0919
	lenient	-3.4354
script	strict	9.4421
	script	**8.5002**
	spirit	8.4568
function	junction	9.9152
	function	**9.8459**
	fraction	-30.9105
	friction	-71.5689
segment	**segment**	**9.5512**
	tensest	8.7067
	tempest	-23.0852

D. Recognition Results of Recurrent Belief Propagation Network

In this section, we list recognition examples generated by our recurrent belief propagation network. Table I lists the top predicted word candidates for some input images. The columns labeled as "Output Word Candidates" and "Excitation Level" give word candidates in the output set with their corresponding excitation level (i.e. score) in descending order. The correct word is highlighted in bold.

Table II lists the recognition for some word images with wrong spellings. These wrong spelled words are randomly selected from [28]. Again, the highest predicted words are reported with their excitation levels listed in descending order. The words in bold font are the actual correct ones.

IV. CONCLUSION AND FUTURE WORK

We have introduced a recurrent belief propagation network for handwriting recognition system. The system construction, processing algorithm and recall process are presented. In our self-structured system, allowing multiple neurons got excited helps improve quality of knowledge link information

TABLE II
WORD CANDIDATES OF SAMPLE WRONG SPELL WORDS

Input Image	Output Word Candidates	Excitation Level
describe	**describe**	**12.3928**
	disperse	11.9450
	perspire	11.8388
effect	effete	12.7191
	effect	**12.6600**
	efface	11.9770
	defect	10.5627
column	colour	11.8852
	column	**9.2361**

and maintain a relatively high accuracy at the same time. Because neurons can retain the knowledge information and reduce the chance of wrong recognition caused by CNN. The proposed preprocessing and post processing techniques effectively reduces the size of predicted word candidates and improves the expected ranking of correct words. The presented belief propagation network is general enough to be applied in other applications for sequence prediction and recall. One of our future work is to extend it for speech recognition. The similar segmentation and deep neural network can be used for pattern matching. A belief propagation network will be constructed with neurons representing phonemes or phoneme sequences and knowledge links between lexicons represent connection between neighboring phonemes. We will also considering other promising methods, such as incorporating higher level context, to improve the accuracy and generality of the framework.

REFERENCES

[1] S. Bhowmik, M. G. Roushan, R. Sarkar, M. Nasipuri, S. Polley, and S. Malakar, "Handwritten bangla word recognition using hog descriptor," in *Emerging Applications of Information Technology (EAIT), 2014 Fourth International Conference of*. IEEE, 2014, pp. 193–197.

[2] M. Dehghan, K. Faez, M. Ahmadi, and M. Shridhar, "Holistic handwritten word recognition using discrete hmm and self-organizing feature map," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 2735–2739.

[3] P. Jifroodian-Haghighi, "A discrete hidden markov model for the recognition of handwritten farsi words," Ph.D. dissertation, Concordia University, 2010.

[4] V. Frinken and S. Uchida, "Deep blstm neural networks for unconstrained continuous handwritten text recognition," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 911–915.

[5] B. Su, X. Zhang, S. Lu, and C. L. Tan, "Segmented handwritten text recognition with recurrent neural network classifiers," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 386–390.

[6] S. Yao, Y. Wen, and Y. Lu, "Hog based two-directional dynamic time warping for handwritten word spotting," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 161–165.

[7] M. Dehghan, K. Faez, M. Ahmadi, and M. Shridhar, "Handwritten farsi (arabic) word recognition: a holistic approach using discrete hmm," *Pattern Recognition*, vol. 34, no. 5, pp. 1057–1065, 2001.

[8] A. Gupta, M. Srivastava, and C. Mahanta, "Offline handwritten character recognition using neural network," in *Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on*. IEEE, 2011, pp. 102–107.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] Y. LeCun, K. Kavukcuoglu, C. Farabet *et al.*, "Convolutional networks and applications in vision." in *ISCAS*, 2010, pp. 253–256.

[11] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in neural information processing systems*, 2009, pp. 545–552.

[12] E. Qaralleh, G. Abandah, F. Jamour *et al.*, "Tuning recurrent neural networks for recognizing handwritten arabic words," *Journal of Software Engineering and Applications*, vol. 6, no. 10, p. 533, 2013.

[13] Q. Qiu, Q. Wu, M. Bishop, R. E. Pino, and R. W. Linderman, "A parallel neuromorphic text recognition system and its implementation on a heterogeneous high-performance computing cluster," *IEEE Transactions on Computers*, vol. 62, no. 5, pp. 886–899, 2013.

[14] Q. Qiu, Z. Li, K. Ahmed, H. H. Li, and M. Hu, "Neuromorphic acceleration for context aware text image recognition," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2014, pp. 1–6.

[15] Q. Qiu, Z. Li, K. Ahmed, W. Liu, S. F. Habib, H. H. Li, and M. Hu, "A neuromorphic architecture for context aware text image recognition," *Journal of Signal Processing Systems*, pp. 1–15, 2015.

[16] N. Yoder, "Peakfinder: Quickly finds local maxima (peaks) or minima (valleys) in a noisy signal," 2014.

[17] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker, "Multi-digit recognition using a space displacement neural network," in *NIPS*, 1991, pp. 488–495.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[19] Wikipedia, "Trie — wikipedia, the free encyclopedia."

[20] "Mieliestronk dictionary." [Online]. Available: http://www.mieliestronk.com/wordlist.html

[21] A. M. Zyarah, "Design and analysis of a reconfigurable hierarchical temporal memory architecture," 2015.

[22] Q. Qiu, Q. Wu, D. J. Burns, M. J. Moore, R. E. Pino, M. Bishop, and R. W. Linderman, "Confabulation based sentence completion for machine reading," in *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on*. IEEE, 2011, pp. 1–8.

[23] Z. Li and Q. Qiu, "Completion and parsing chinese sentences using cogent confabulation," in *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2014 IEEE Symposium on*. IEEE, 2014, pp. 31–38.

[24] Z. Li, Q. Qiu, and M. Tamhankar, "Towards parallel implementation of associative inference for cogent confabulation," in *High Performance Extreme Computing (HPEC), 2016 IEEE Conference on*. IEEE, 2016.

[25] "Machine learning and pattern recognition density estimation: Gaussians." [Online]. Available: http://www.inf.ed.ac.uk/teaching/courses/mlpr/lectures/mlpr-gaussian.pdf

[26] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images." in *VISAPP (2)*, 2009, pp. 273–280.

[27] F. Pereira, M. Riley, and R. Sproat, "Weighted rational transductions and their application to human language processing," in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994, pp. 262–267.

[28] "240 common spelling mistakes in english." [Online]. Available: http://www.engvid.com/english-resource/common-spelling-mistakes-in-english/