Handwriting Recognition with Beam Search Decoding

Zixuan Li
University of Southern California, California, United States
{zixuanli}@usc.edu

Abstract: Neural networks (NN) consisting of convolutional NN layers and recurrent NN layers combined with a final connectional temporal classification (CTC) layer is helpful for Handwritten Text Recognition (HTR). In this report, I will introduce potential uses and current challenges in handwritten recognition. I'll then review related previous publications on handwriting recognition and implement my own Neural Network (NN) model with two decoding techniques, best path decoding and vanilla beam search decoding. This report will discuss how beam search decoding works better than the best path decoding. A quantitative analysis is given for the IAM dataset. A trained HTR system is coupled with both algorithms and the error rates are compared.

1 Introduction

Confronting with unrelenting changes in society, Handwriting Text Recognition (HTR) is gaining increasing attention. It is important because it can be used in various industries to reduce labor costs and save precious man-hours. The global Optical Character Recognition (OCR) Market size is expected to reach USD 13.38 billion by 2025, expanding at a CAGR of 13.7% from 2019 to 2025 due to rising investments in OCR startups and adoption of OCR across various industry verticals [1]. There are still challenges out there for HTR even though OCR has been considered as a solved problem. It suffers from people's poor handwriting and high variances of handwriting styles.

The objective of this report is to briefly review previous publications on HTR and then compare best path decoding and beam search decoding and give suggestions on which algorithm is better to use. The comparison is done regarding the concepts of the algorithms are based on, their accuracy and error rate for the IAM dataset.

2 Use Cases and Challenges

Handwriting Text Recognition (HTR) is commonly used in our daily lives. It is crucial to help with diversified industries such as healthcare, insurance, and banking.

2.1 Healthcare

Prescription digitization is essential in the healthcare/pharmaceutical industry. In some countries, for example, China, doctors have special handwriting styles that are barely recognizable by patients. This situation worsens especially when doctors are using abbreviations [2] while patients are not familiar with their prescription and reluctant to ask for clarification (see Figure 1). By adding handwriting recognition to their toolkit of services, hospitals/pharmacies can significantly improve customer experience.

2.2 Insurance

A large insurance industry receives millions of documents a day and a delay in processing the claim can impact the company terribly. The claims documents can contain different handwriting styles, some of which are hardly recognizable, which will slow down the pipeline if all relying on manual processing.

2.3 Banking

People write checks on a regular basis. Usually, the check procedure requires a bank employee to read and manually enter the information present on the check and verify the entries including signature, date, and amount. As a significant amount of check being process every day, a handwriting recognition system can not only save labor costs and hours of human work, also customers can deposit their checks without going to the bank. For example, Chase online deposit gives users a choice to deposit checks online even though it still cannot recognize users' handwriting well.

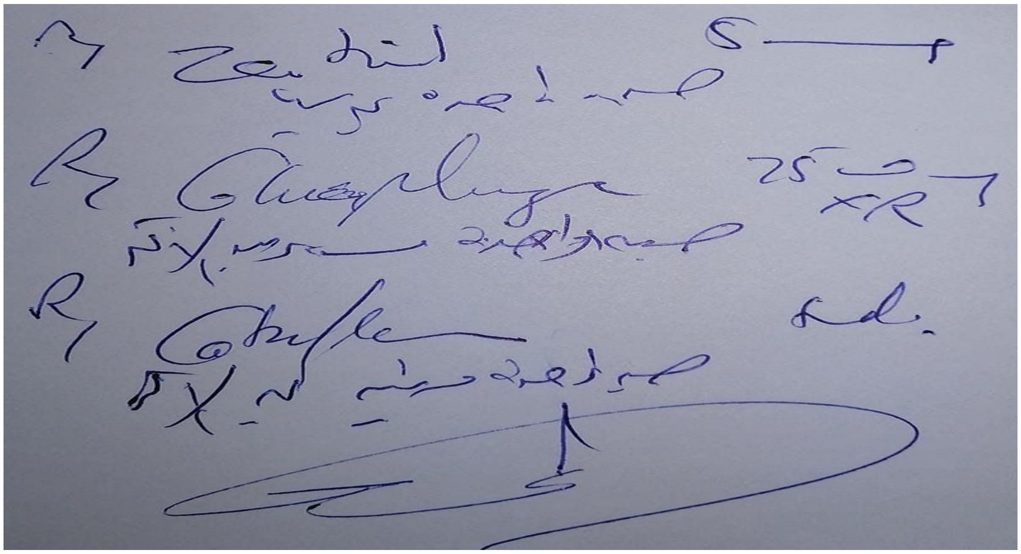


Figure 1: A prescription written in Arab.

2.4 Challenges in HTR

1. The handwriting style of an individual varies from time to time
2. Huge variability and ambiguity of strokes from person to person
3. People tend not to write in a straight line on paper
4. Cursive and blurry handwritings
5. Difficult to collect a well-labeled handwriting text dataset

3 Related Work

After reviewing past publications related to RNN/LSTM and various decoding methods, I choose to implement a multi-dimensional recurrent neural network and compare results on two different types of CTC decoding methods: best path decoding and vanilla beam search decoding.

First, how does CTC work? CTC allows training text recognition systems with pairs of images and ground-truth texts [3]. Text is encoded in the NN output matrix by paths, which contains one character at each position. Each character of a text can be repeated multiple times and an arbitrary number of CTC blanks is inserted between characters (at least one blank must be inserted in the case of repeated characters). Then we sum over the probabilities of all corresponding paths and select a certain number of paths with the highest probabilities we want.

3.1 Best Path Decoding

This is the result shown in Figure 2. At each position, we take the output of the NN and select the character with the highest probability to compute the best path, then remove duplicate characters and CTC-blanks from this path [4].

3.2 Vanilla Beam Search Decoding

It also only uses the NN output but it uses more information from it and produces a more accurate result. Instead of taking a single output from NN, it iteratively generates labels and keeps the best k labels all the time, where k denotes the beamwidth.

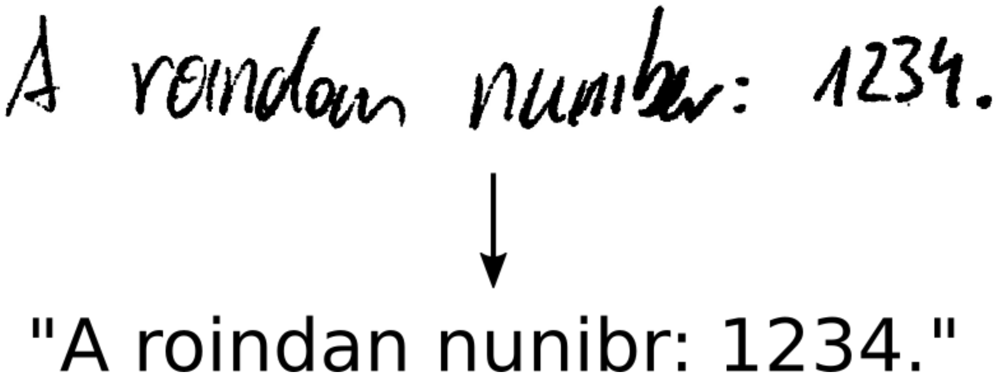


Figure 2 (Source: https://towardsdatascience.com/word-beam-search-a-ctc-decoding-algorithm-b051d28f3d2e).

4 Models

The NN model [5] I'll use depends on NumPy, cv2, and TensorFlow imports. It consists of 5 CNN layers, 2 RNN (LSTM) layers and the CTC loss and decoding layer. The workflow includes 3 steps. **Step 1**: The CNN layers extract the visual features which are passed to the RNN layers. The output is a feature map. **Step 2**: Two LSTM layers propagate information through the sequence and map the sequence to a matrix of size 32x80 (as illustrated in Figure 3). **Step 3**: The CTC layer either calculates the loss value (when training) or decodes the matrix to the final result with two decoding algorithms.

4.1 Implementation of Best Path Decoding

As shown by Graves [4], we first get char indices along the best path. Then we collapse the best path, turn indices back to characters and join the characters to get the text string. Since we only need to find the character with the highest score for each time-step, the best path decoding takes little time. Suppose we have C characters and T time-steps, the runtime for best path decoding should be O(C · T).

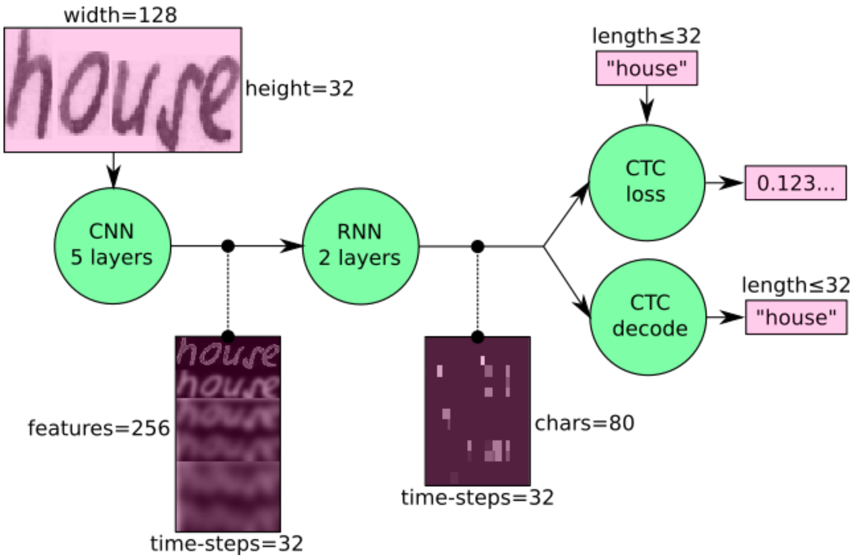


Figure 3: Architecture for an HTR system.

4.2 Implementation of Beam Search Decoding

Hwang introduces beam search decoding in his paper [6]. This algorithm takes in RNN output, beam width *bw* and a language model P and outputs the most probable labeling. It iterates through the NN output and creates labelings (called beams) which are scored. At each time-step, the beams in the set are extended by all possible characters, forming a tree of labelings as shown in Figure 4. As the number of vertices of this tree grows exponentially in time, only the *bw* best beams are kept to keep the algorithm feasible and all others are removed. Equal labelings get merged.

Take Figure 4 as an example. We start with a blank which corresponds to the root of the tree. The beam is then copied and extended by all possible characters {"a", "b", "-"}. In level 1, we only keep the 2 best beams so we throw away the beam "b" which has a probability of 0. We extend the tree again with characters in the alphabet and get "aa", "ab", "a", "a", "b", "-". We now merge two "a"s by adding up their probabilities and only keep one of them. Again we remove all beams containing "b" since "b" has 0 probability. We also remove "aa" because to encode a text with duplicate characters, we have to insert a blank in between, which is impossible for a path of length 2 so "aa" has 0 probability. Now what remains are the beams "a" and "-" with probability 0.52 and 0.48 respectively. Thus, we finish the last iteration and return the beam with the highest probability, which is "a".

5 Results

I'll first compare the implementation between two decoding methods intuitively. Then I'll evaluate the performance with the trained NN on the IAM handwriting database [7].

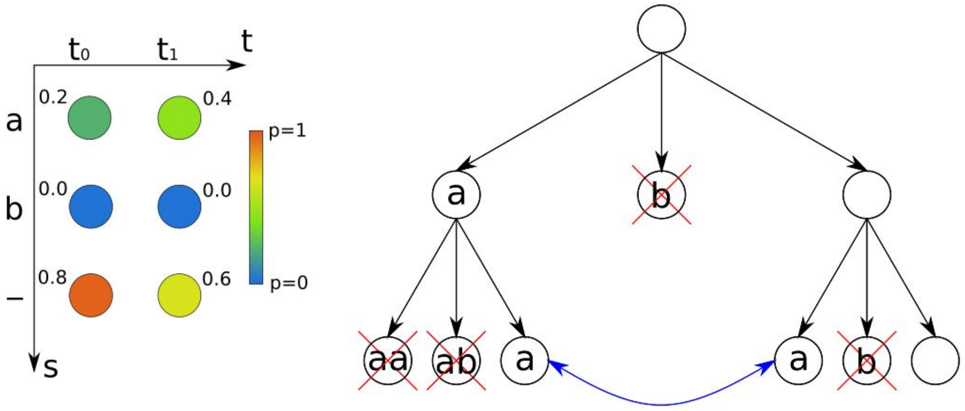


Figure 4 [3]: NN output and tree of beams with alphabet = {“a”, “b”} and BW = 2.

5.1 Conceptual Comparison

Best path decoding is faster but less accurate than the beam search decoding. It runs faster because it only needs to find the character with the highest probability score for each time-step and the probability of the path is the multiplication of the probabilities of each character. However, it may fail in certain situations like Figure 5.1 when the blank spaces are selected while there are other paths (eg. path for the text “a” in Figure 5.2) with larger probabilities.

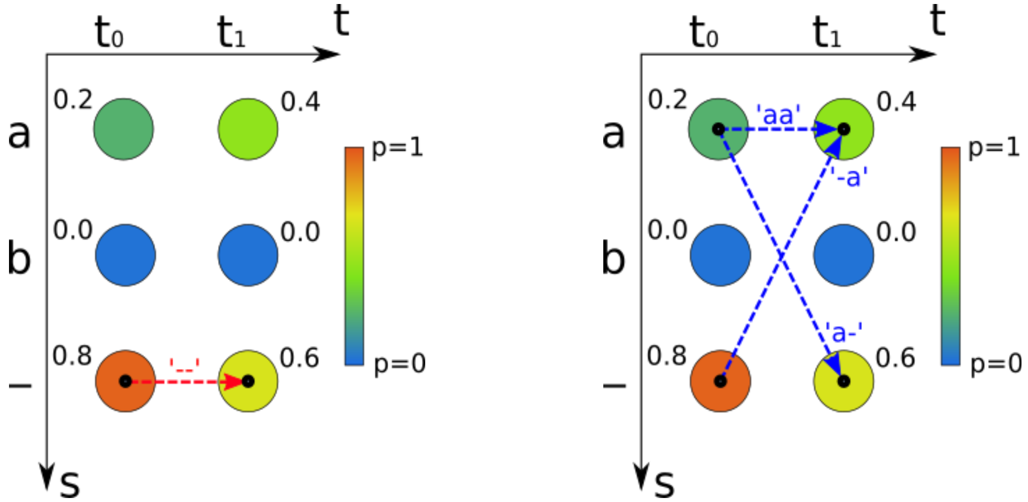


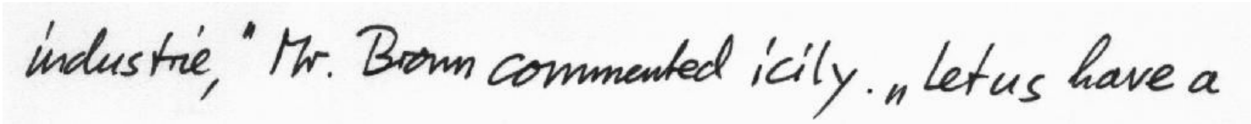
Figure 5 [3]: (1) Select characters with the highest probability. (2) All paths corresponding to text “a”.

5.2 Evaluation

The algorithms are compared using the IAM dataset [7] (a sample is shown in Figure 6) with 95% of the dataset as the training set (109552) and 5% as the validation set (5766). The IAM dataset I’m using contains forms of unconstrained handwritten English text which were scanned at a resolution of 300 dpi and saved as PNG images with 256 gray levels.

To evaluate the performance of the algorithms, Character Error Rate (CER) and Word Accuracy Rate are used as error measures. The CER is the Levenshtein distance between the ground-truth text and the recognized text, normalized by the length of the ground-truth text. The same applies to WER but on a word-level.

The results are shown in Table 1. As can be seen, beam search decoding outperforms best path decoding for both CER and WER. This is expected since beam search encoding correctly incorporates multiple paths into the result instead of greedily selecting the path as the best path decoding does.



industrie," Mr. Brown commented icily. " let us have a

Figure 6: A sample from IAM dataset [7].

Decoding Algorithms	Character Error Rate (CER)	Word Error Rate (WER)
Best path decoding	13.956289%	32.278261%
Beam search decoding	10.361747%	25.930435%

Table 1: Result for the IAM dataset.

6 Discussion

6.1 Lessons Learned

I ran the NN on my custom image of handwritten text but it gave me the wrong result no matter how I tried. Turns out that it failed because the NN is trained on the IAM dataset and it learned properties of the dataset-images, be they cropped text, high contrast, and mostly lower-case characters, at the same time it learned to recognize texts. It might work if I trained the NN with a custom handwritten dataset but it is too difficult to generate a huge amount of handwritten texts.

6.2 Potential Optimization

When I did the research, I noticed that there are other decoding algorithms like word beam search which improves the accuracy of beam search and token passing that looking for the most probable word sequences. Among them, token passing can achieve better results if all words to be recognized are known in advance. Word beam search seems to be more advanced than the best path search and beam search and outperforms both of them on the IAM dataset. However, due to some inevitable technological issues, I was not able to run the model with the word beam search algorithm. It definitely worths trying if I have a chance in the future.

7 Conclusion

In this report, I firstly gave an introduction to why HTR is an interesting and useful topic, including possible uses and current challenges in HTR. Then I explained how CTC works and talked about two CTC decoding methods: Best path decoding and beam search decoding. Best path decoding is a fast and simple approximation and it selects the character with the best score greedily. Beam search decoding uses a character-level LM and outperforms the best path search on both CER and WER on the IAM dataset.

This report demonstrates that beam search decoding is a more accurate decoding algorithm compared to the best path decoding though it takes a longer execution time. Therefore, if speed matters, best path decoding is more well-suited than beam search decoding. Otherwise, beam search decoding is the better choice.

References

[1] “Optical Character Recognition Market: OCR Industry Report, 2019-2025.” *Optical Character Recognition Market Size, Share & Trends Analysis Report By Type (Software, Services), By Vertical (Retail, BFSI, Government, Education, Healthcare), By Region, And Segment Forecasts, 2019 - 2025*, July 2019, www.grandviewresearch.com/industry-analysis/optical-character-recognition-market.

[2] Al-Ghannam, Talal. “Doctor's Scribbles A Puzzle Understood by 'Pharmacists'.” *Doctor's Scribbles A Puzzle Understood by 'Pharmacists'* | *MENAFN.COM*, 7 May 2019, menafn.com/1098489999/Doctors-scribbles-A-puzzle-understood-by-pharmacists.

[3] Scheidl, Harald. “Beam Search Decoding in CTC-Trained Neural Networks.” *Medium*, Towards Data Science, 30 June 2020, towardsdatascience.com/beam-search-decoding-in-ctc-trained-neural-networks-5a889a3d85a7.

[4] Graves, Alex. "Supervised sequence labeling." *Supervised sequence labeling with recurrent neural networks*. Springer, Berlin, Heidelberg, 2012. 5-13.

[5] Scheidl, Harald. “Build a Handwritten Text Recognition System Using TensorFlow.” *Medium*, Towards Data Science, 9 Aug. 2020, towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5.

[6] K. Hwang and W. Sung. Character-level incremental speech recognition with recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5335-5339. IEEE, 2016.

[7] U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002.