Handwritten Editor

Hidehisa Arai
TUM Department of Informatics, Chair for Applied Software Engineering
Technical University of Munich
Munich, Germany
koukyo1213@g.ecc.u-tokyo.ac.jp

***Abstract*—Handwriting is still in strong demand as a way of communication and saving knowledge even in the era of new technology. Driven by the rise of electronic pens and tablets, the need for handwriting recognition on electronic devices is increasing recently.**
This report focuses on handwriting recognition on iOS device and discusses how to use the recognition results using auto-completion as an example. Since handwritten document recognition technology is mature, this report focused more on reducing the amount of computation for on-device inference.

I. Introduction

Handwriting recognition (HWR) is a field which study and develop algorithms to interpret handwritten inputs into a format that can be easily handled by computers from sources such as papers, photographs, electronic tablets, and other devices. HWR can be roughly devided into two appproaches: online approach and offline approach [1].
While online approach uses information on the trajectory of the pen tip obtained from a special pen for classification, offline method uses optically scanned images as input and performs recognition using computer vision techniques. This report focuses only on offline approach, and tackle on the recognition problem using Convolutional Neural Network (CNN), which have shown a remarkable development in recent years.

HWR is a field that has been studied for a long time, and many applications have already been made. However, in the case of tablet devices that have spread rapidly in recent years, not so many applications have been created even after APIs to incorporate with pattern recognition algorithms on device are published by developpers of those devices. Although some applications[1] have achieved very good result on normal handwritten text recognition, it is not the case when elements in other domains such as handwritten illustlations are mixed in addition to sentences. Also, APIs published by the manufacturer of those devices are usually black-boxes and poorly extensible. The main target of these APIs is printed text, and when applied to handwritten characters, performance may not be as high as expected.

I therefore endeavor to recognize handwritten documents on device, especially those which contain not only text, but also handwritten illustlations or mathematical formulas. Since this type of documents are very common in our daily life, the success of the project can potentially bring the fusion of digital technology and the long-standing human skills of handwriting.

In the following section, typical approaches of HWR and related works are introduced. Section three provides technical details of the approach to the problem addressed in this report. Section four describes the result of the approach and discuss on that. Section five concludes this report with future prospects.

II. Related Work

The problem settings of this project can be positioned as one variant of Scene Text Detection/Recognition, which is a field to study algorithms to extract and recognize text information written in natural images. Due to the recent development of Neural Networks technology, much research has been done in this field to this day [2].
Except for few methods [3] [4], most approaches of Scene Text Detection/Recognition separate detection and recognition and perform stepwise inference.

A. Detection

Scene Text Detection can be subsumed under general object detection, therefore those methods usually follow the same procedure of object detection, which is dichotomized as one-stage methods and two-stage ones [5].

After the emergence of FasterR-CNN [6], most of the modern text detection algorithms are based on FasterR-CNN, YOLO [7], SSD [8]. In addition to the general object detection model, text detection models are devised to detect tilted bounding boxes and character regions of arbitrary shapes, or to simplify the pipeline.

Since this report does not elaborate on text detection with the reasons described below, no more detailed explanation are provided.

B. Recognition

Some text recognition algorithms devide the task into character segmentation and character recognition [9] [10]. Character segmentation is considered as the most challenging part of scene text recognition, and may affect overall accuracy. It is especially difficult to segment connected characters such as cursive. Therefore some techniques which do not rely on character segmentation have been developped so far. This

[1] https://www.nebo.app/

report introduces a method called Connectionist Temporal Classification (CTC) [11].

CTC was first introduced to handle sequence labeling of arbitrary length, requiring no pre-segmented training data. A CTC network outputs probabilities for each label at each time step. Time step length can be any length longer than label length. The output at each time step is the probability of the classess to be recognized plus the extra class representing "blank". Let this output probabilities be $\mathbf{y} = (y_1, y_2, \cdots, y_w)$ and denote by $y^t_{\pi_t}$ the activation of label π_t at time step t. Given this probability distribution, the conditional probability of the sequence is calculated as follows.

$$p(\pi|\mathbf{y}) = \prod_{t=1}^{w} y^t_{\pi_t} \tag{1}$$

Then a many-to-one mapping $\mathcal{B}$ is defined to transform the sequence π to a shorter sequence. The final predicted label is obtained by this mapping. This mapping removes all blanks and repeated continuous labels from the sequence. For example, $\mathcal{B}$ maps the predicted sequence "aa-p-pl—-ee" to "apple", where "-" represents the "blank". Since this mapping is many-to-one mapping, different sequences may be mapped to the same sequence. Therefore the probability of the final output sequence is the sum of all possible conditional probabilities of all π corresponding to that final sequence.

$$p(l|\mathbf{y}) = \sum_{\pi} p(\pi|\mathbf{y}) \tag{2}$$

where π represents all π which produces $l = \mathcal{B}(\pi)$.

The output of the classifier should be the most probable labeling for the input sequence.

$$h(\mathbf{y}) = \arg\max p(l|\mathbf{y}) \tag{3}$$

In general, there are a large number of mapping paths for a give sequence, thus calculation of $\arg\max$ requires heavy computation. In practice, following two approximate methods are known to give us a good result.

The first method is based on the assumption that the most probable path can be approximated by the sequence of most probable labeling

$$h(\mathbf{y}) \approx \mathcal{B}(\pi^*) \tag{4}$$

where π^* is a set of labels which get the highest probabilities at each time step. Although it works well, it is not guaranteed to get the most probable labeling.

The second method is to use forward-backward algorithm to efficiently search for the most probable sequence. With enough time, this approach can always find the most probable labeling from the input sequence, but the amount of computation increases exponentially with respect to the sequence length, it is not practical to find the exact solution.

To train the network with the dataset $\mathcal{D} = \{I_i, l_i\}$, where I_i represents the input image and l_i represents the corresponding label, maximum likelihood approach it utilized. The objective function of this can be negative log-likelihood

$$\mathcal{O} = -\sum_{(I_i,l_i)\in\mathcal{D}} \log p(l_i|\mathbf{y}_i) \tag{5}$$

where $\mathbf{y}_i = f(I_i)$ and $f(\cdot)$ represents the classifier. To minimize negative log-likelihood, Stochastic Gradient Descent (SGD) can be used.

III. Methods

This section describes in detail the approach to the classification problem in documents that contain handwritten text as well as handwritten illustrations.

As in the case of Scene Text Detection/Recognition, it is effective to separate Text Detection and Text Recognition and treat them as different problems. Furthermore, it is possible to record the written area due to the characteristics of the electronic tablet, so using a simple heuristic on the trajectory data eliminates the need to actually perform text detection. The role of this step, namely "Region of Interest Detection" step is to reduce the size of data passed to the subsequent processing and suppress the increase in the amount of calculation.

In general, electronic tablets have severe limitations on computing power, so in this report I used the heuristic to detect region of interst. Detected regions are then preprocessed and passed to the text recognition module. In the text recognition module, two patterns of recognition using CTC and recognition using the API provided by Apple were verified.

A. Region of Interest Detection

The purpose of region-of-interest detection is to cut out subsequence representing words and illustrations from the dot sequence of the pen tip trajectory and cut back the data to be passed to the subsequent processing to reduce the amount of calculation and improve the recognition accuracy. Region of interest is a region including a dot sequence of the word or the illustration. To specify the region, two assumptions are made on region of interest.

1) Objects drawn in the region of interest are close in time when drawn
2) Objects drawn in the region of interest are spatially close

Based on these assumptions, region of interest is successfully detected. Figure 1 shows an example of the region detected with this heuristic.

There are some cases where such heuristics do not work. For example, when the scale of the depicted object is large, the gap between the sequence of points constituting the object may be too large and fail.It is also a major problem that temporal and spatial proximity depends on parameters and sometimes does not match intuition. However, it is certain that this method can narrow down the target area for text recognition with a very small amount of computation, so this report adopted this method.

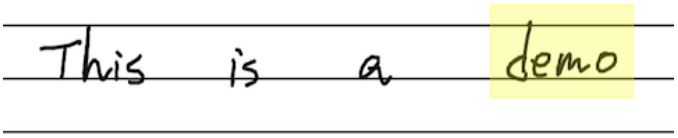

Fig. 1. The region with yellow overlay is detected region of interest

B. Recognition

In text recognition, two models were tried: a model that directly reads the content from the result of region of interest detection using CTC, and the model which is provided by Apple[2]. Note that Apple's VNRecognizeTextRequest API does not disclose its implementation or the model used inside, it is unknown exactly what processing is being done.

1) CTC: CTC models are usually constructed using CNN model with Recurrent Neural Network (RNN) as the output layer. However, RNN performs computations slowly compared to CNN, and in recent years its effectiveness in CTC is sometimes questioned [12]. Given that all calculations are done on iOS, and based on research that the output layer of the CTC model can also be configured using CNN [13], in this report a model which is fully constructed only by CNN was used for CTC model. To get an approximation for most probable labeling, greedy policy, the policy to take the most probable label at each time step, was used.

2) VNRecognizeText API: VNRecognizeText API is an API provided by Apple officially, and its function is to detect text from the whole area where the drawing is and read it. Therefore there's no need to detect the region of interest to read the content inside. This API is completely a black box, accepts an image of any shape as input, reads the text in it, and outputs it with position information.

VNRecognizeText API has two modes for recognition, namely '.fast' and '.accurate'[3]. According to the official introduction video[4] if VNRecognizeText API, '.fast' uses traditional feature based method inside while '.accurate' uses more sophisticated Computer Vision model inside. As the names of those imply '.fast' is faster but not so accurate while '.accurate' is slower but more accurate. Both '.fast' and '.accurate' were tested. However, since '.fast' failed to recognize any handwritten letters, measurement on '.fast' was not conducted.

C. Auto-Complete

In this report, I deal with auto-completion as an application of the inference result. This auto-completion emphasizes simplicity and presents words that start with infered result in order of frequency. To get the word candidates which start with infered result, I created an inverse index that maps the first few letters of words to a group of words with that prefix. For example, index "elbo" corresponds to word group {"elbow", "elbow's", "elbowed", "elbowing", "elbowroom", "elbowroom's", "elbows"}. Word suggestions for auto-completion are from /usr/share/dict/words of Debian GNU/Linux. Word frequencies were counted using wikipedia-word-frequency[5].

[2]https://developer.apple.com/documentation/vision/vnrecognizetextrequest
[3]https://developer.apple.com/documentation/vision/vnrequesttextrecognitionlevel
[4]https://developer.apple.com/videos/play/wwdc2019/234/

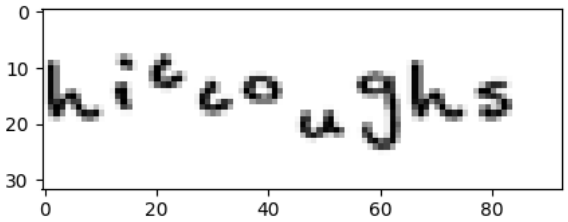

Fig. 2. Generated image using handwritten-like fonts

D. Dataset

Handwritten character recognition and handwritten sentence recognition are fields that have been studied for a long time, so there are many data sets, but these are often provided in different formats, and there is some difficulty in eliminating differences between formats and using them for training dataset.

I therefore took an approach to create a composite dataset by embedding a combination of existing handwritten-like fonts and randomly selected English words in the image. Figure 2 shows an example of training data generated with this method.

E. Implementation

I used Python3.6[6] and TensorFlow2.1[7] for building machine learning models and training, and used coremltools3.2[8] to convert the models into the format that can work on iOS.

To get handwritten document image from the iOS electronic tablet, a function to print the input from the Apple Pencil[9] on the white canvas according to the path of the pen has been implemented.

For training the model, Google Colaboratory environment [10] has been used. All the implementation is published at the GitHub repository[11].

IV. Results & Discussion

A. Region of Interest Detection

It is desirable for region of interest detection to cut out the region which if focused by the user. For example, when a user is writing a sentence,region of interest should be a word written, and when a user is drawing an illustration, region of interest should be the whole illustration drawn.

[5]https://github.com/IlyaSemenov/wikipedia-word-frequency
[6]https://www.python.org/downloads/release/python-360/
[7]https://www.tensorflow.org/
[8]https://apple.github.io/coremltools/
[9]https://www.apple.com/apple-pencil/
[10]https://colab.research.google.com/notebooks/welcome.ipynb
[11]https://github.com/koukyo1994/iOS-note-v2

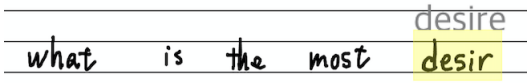

Fig. 3. region of interest detection works well when cutting out a word which is being written

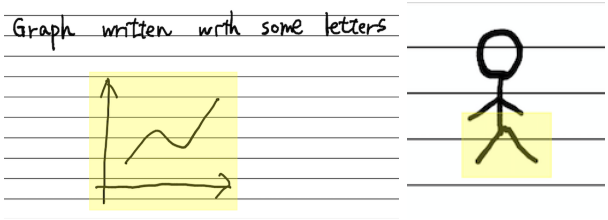

Fig. 4. Left: example of the case ROI detection worked well on illustration. Right: example of failure case of ROI detection

Since it is difficult to quantitatively measure the goodness of the region of interest detection, only qualitative evaluation was performed this time.

When cutting out only the words that the user is writing in the text, the region of interest detection worked almost without failure (Fig 3).

On the other hand, when cutting out a handwritten illustration, only a part of the illustration may be cut out if the lines constituting the illustration are not spatially close to each other (Fig 4).

B. Evaluation of Text recognition

In the previous section, two patterns of text recognition were tried: recognition using CTC and recognition using VNRecognizeText API. Since this report worked on recognition of handwritten text and made use of the result for auto-complete, following metrics were designed to measure the goodness of auto-completion.

- Omitted Characters Count (OCC) - The gap between how many characters did the writer actually write before he found the word he wanted to write within top 10 of the auto-completion candidates and the number of characters in the word. If it is not found after writing to the end, the score is 0. Higher value of this metric indicates better result.
- Cumulative Time for Inference (CTI) - Cumulative time spent on auto-completion until the word that the writer actually wants to write is included in the top 10 auto-completion candidates. If it is not found after writing to the end, the score cumulative time spent on the recognition at each step. Note that each time writer releases the pen tip from the tablet, recognition process runs. Lower value of this metric indicates better result.

100 words were selected from the word list in /usr/share/dict/words of Debian GNU/Linux for evaluation. Table I shows the performance of both methods on auto-completion. While OCC measure of VNRecognizeText API show better result than CTC, cumulative time elapsed for inference is almost doubled compared to that of CTC's.

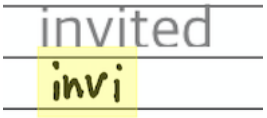

Fig. 5. Example of auto-completion showing the top 1 candidate

Fig. 6. Example of unstable inference

method	OCC (mean)	CTI (mean)
CTC	1.0102	1.1060
VNRecognizeText (.accurate)	3.3405	1.9751

TABLE I
PERFROMANCE OF CTC AND VNRECOGNIZETEXT API ON AUTO-COMPLETION.

Figure 5 show an example of how auto-completion works.

C. Discussion

The VNRecognizeText API is superior to CTC in character recognition accuracy, but in terms of speed it takes about three times longer to infer. Since auto-completion is an application that requires real-time performance, a small lag does not provide a good user experience. Therefore, poor performance in speed can be a problem.

On the other hand, the CTC model shows inference speed that does not the user feel uncomfortable in actual use, but the inference result is unstable, and a slight difference in notation greatly affects the inference result. Figure 6 shows an example of unstable inference.

This is probably because training of the CTC model seems to be over-fitty and strongly depends on the dataset used for training. Therefore, future tasks include reducing the reliance on datasets and using larger datasets or using data augmentation techniques to improve generalization performance.

V. FUTURE WORK & CONCLUSION

The goal of this report is to create an application to recognize sentences on iOS for documents where handwritten illustrations and characters were mixed. Although it is imperfect, the heuristic used for region of interest detection successfully detect a word out of a sentence, and can cut handwritten illustration out from the other part of document without requreing large amount of computation. On the other hand, text recognition had a trade-off between speed and accuracy.

Future tasks include improving the accuracy of text recognition without slowing it down. Another example of future work

is to perform more accurate text recognition and ROI detection using an online method. In auto-completion, prediction using information on surrounding words, and refinement of the method of presenting complement candidates can also be one of future works.

REFERENCES

[1] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[2] S. Long, X. He, and C. Yao, "Scene text detection and recognition: The deep learning era," 2018.

[3] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "Fots: Fast oriented text spotting with a unified network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5676–5685.

[4] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai, "Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–83.

[5] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *arXiv preprint arXiv:1809.02165*, 2018.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[9] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "Photoocr: Reading text in uncontrolled conditions," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 785–792.

[10] T. Q. Phan, P. Shivakumara, B. Su, and C. L. Tan, "A gradient vector flow-based method for video character segmentation," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 1024–1028.

[11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[12] J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?" in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 67–72.

[13] Y. Gao, Y. Chen, J. Wang, and H. Lu, "Reading scene text with attention convolutional sequence modeling," *arXiv preprint arXiv:1709.04303*, 2017.