Optical Character Recognition (Handwritten) With Modern Neural Networks

Siqi Chen
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
`carasiqi.chen@mail.utoronto.ca`

Yun Qi Fang
Department of Computer Science
University of Toronto
Toronto, ON, M5S 1A1
`yunqi.fang@mail.utoronto.ca`

Zheyu Yang
Department of Computer Science
University of Toronto
Toronto, ON, M5S 1A1
`zheyu.yang@mail.utoronto.ca`

Abstract

In this paper, we study the architectures of recognizing characters that are specifically handwritten. Vertical Attention Network (VAN) and Transformer are two methods/models that are recently new introduced in optical character recognition and have achieved outstanding accuracy on the task. In this paper, we will explore these two new models by analyzing the models and tuning the hyper-parameters used in the model. We will also identify the limitations of each of the models and explore how they can be applied to handwritten text recognition.

1 Introduction

By 2025, the market for optical character recognition (OCR) is estimated to be worth USD 13.38 billion, up 13.7 percent year over year. This expansion is fueled by the rising digitization of corporate operations, which uses OCR to save labor costs and save valuable man-hours. Although OCR has been deemed a solved problem, one of its important components, Handwriting Recognition or Handwritten Text Recognition (HTR), remains a difficult challenge to tackle. Converting handwritten text to machine-readable text is difficult due to the wide range of handwriting styles across people and the low quality of handwritten text compared to printed text. Nonetheless, it's a critical issue for a variety of businesses, including healthcare, insurance, and banking. While Google and Microsoft may have more professional HTR services/APIs; in this paper, we also want to try to re-implement a version with techniques we learned in the courses.

2 Related Works

2.1 Hidden Markov Model (HMM) & Vertical Attention Network (VAN)

Since 1998, Amlan Kundu et al. have conducted a series of in-depth experimentation on different variations of HMM to determine differences between previous models under distinct cases of limitations [1]. While HMM is famously known for its readability, relatively efficient learning algorithms and state-of-art accuracy, it is critiqued for the Markov assumption itself, which states that the probability dependence only exist between the current time instance and the immediate previous time instance; this, does not always stay true as the soundness of sentences can still be held across multiple states

[5]. There are two approaches based on HMMs where the early one learning the states over small sets of letters which is more applicable and stable for smaller domains and another approach introducing letters as states which has higher applicability and the model itself becomes less complex [6]. In 2022, D. Coquenet et al. introduced a Vertical Attention Network (VAN) to solve HMMs' limitations when dealing with long term dependencies.[8]

2.2 Transformer-based Optical Character Recognition(TrOCR)

Transformers, which is a recurrence-free architecture becomes a rising star in OCR tasks very recently. D.H. Diaz et al. from Google, conducted a study on general text line recognization by comparing different combination of encoder and decoder. The Transformer decoder with self-attention module shows a decent performance in the study [7]. Also, R. Atienza proposed a new model architecture ViTSTR, that is built on top of the vision transformer, and applied in Scene Task Recognition (STR). ViTSTR contains only one stage of Transformer encoder, so it computes faster and less cost than other transformer-based STR architecture, but with competitive accuracy [9]. Also, C. Wick et al. proposed a Transformer-based encoder/decoder with bidirectional decoding, using Convolutional Neural Network(CNN) for feature extraction [10]. In 2021, M. Li et al., proposed TrOCR, a Transformer-based OCR model for text recognition with pre-trained Computer Vision model as the encoder and Natural Language Processing model as the decoder, without CNN as the backbone[11].

3 Methods

3.1 IAM Handwriting Database

IAM Handwriting Database is a unconstrained handwritten English database that contains sample handwriting from 657 writers, 1539 pages of scanned text, consists of 5685 isolated and labeled sentences in total. The database has been extensively used in writer independent OCR tasks in many researches and publication.

3.2 Data preparation

The input images of each dataset are downscaled by a factor 2 through interpolation and then padded to reach a minimum height of 480px and a minimum width of 800px. Data augmentation (ex. resolution modification, perspective transformation, elastic distortion and etc.) is also applied to each dataset to reduce over-fit and improve stability at the training time.

3.3 Vertical Attention Network (VAN)

VAN is an end-to-end model which takes a Fully Convolutional Network (FCN) encoder, an attention module and a decoder and is trained by a combination of two losses which are the connectionist temporal classification (CTC) loss and the cross-entropy loss.
A FCN encoder which is composed with 6 Convoluition Blocks and 4 Depthwise Separable Convolution Blocks is implemented to extract features from the input mages. Once the feature f has computed from the encoder, a hybrid attention mechanism computes attention weights from both the content and the location output layers.
The attention weights a_t (t means process of the t_{th} line) are computed as follows:

$$s_{t,i} = \tanh(W_f \cdot f'_t + W_j \cdot j_{t,i} + W_h \cdot h_{W_f \cdot (t-1)})$$
$$e_{t,i} = W_a \cdot s_{t,i}$$
$$a_{t,i} = Softmax(e_{t,i})$$

where i represents each row, $s_{t,i}$ contains information from the current feature and previous attention weights, W_f, W_j, W_h are weights of connected layers, f_i is the feature extracted from the i_{th} line, $j_{t,i}$ represents the collected information from the past attention steps and $h_{W_f \cdot (t-1)}$ which represents the content-based part is the hidden state after processing the $i-1$ line, $e_{t,i}$ is the score computed for line i and W_a are the weights of a connected layer. Then the scores are computed through the softmax activation to get the final attention weights a_t. The decoder is applied by a LSTM layer and the final loss is computed by the addition of the CTC loss and the cross-entropy loss.
See Appendix A Figure 1 for VAN model architecture.

3.4 Transformer-based Optical Character Recognition(TrOCR)

TrOCR is an end-to-end Transformer-based model that uses a image Transformer as the encoder and text Transformer as the decoder.
The image Transformer encoder takes in the input image and resize it to a fixed size (H, W). Then they split the image in separate patches, with size (P, P), where H and W can be evenly divided by P. The Transformer has a hidden size D, then every patch will be projected into a D-dimensional patch embedding, and feed into the Transformer. Each group of encoder layer consists of 1 feed forward network and 1 multi-head attention layer, both are followed by residual connection and layer normalization, and there are N stacks of them. The multi-head attention is computed as follow:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{where } Attention(Q, K, V) = softmax(\frac{QK^T}{d_k})$$

Here, Q is queries, K is keys and V is values, and d_k is dimension of queries and keys.

For the decoder, it also contains stacks of identical layers, which each contains a feed forward layer, multi-head attention layer and a masked multi-head attention layer. The output for the i-th position can only look at the inputs before i-th position, because of decoder output will right shift one position from the decoder input.
The probability is calculated by:

$$h_i = Proj(Emb(Token_i))$$
$$softmax(h_{ij}) = \frac{e^{h_{ij}}}{\sum_{k=1}^{V} e^{h_{ik}}} \text{ for } j = 1, ..., V$$

Here, V is the vocabulary size. Then use beam search to get the final output.

See Appendix A Figure 2 for TrOCR model architecture

4 Experiments and Discussion

To compare and contrast the overall performance of VAN and TrOCR, we experimented them to their fullest potential with respective data sets. We will present several aspects including fine tuning hyper-parameters, accuracy and efficiency comparison table, and some custom samples we created.

4.1 Limitations

Due to limited computing power of GPUs we possess, CUDA may only contain training data when limiting the number of epochs and batch size to be reasonably diminutive, which could imply a limited statistical representativeness, however, the results could suffice comparisons.

4.2 Fine Tuning Hyper-parameters

4.2.1 VAN - Hidden sizes

The default hidden size for VAN set by D. Coquenet et al.[8] was 256, however, by experimenting different sizes (64, 128, 512 etc.), the difference between resulting accuracy and time cost varies. Generally, assuming reasonable sizes are given, increasing hidden size will increase the time required to train each epoch; additionally, the accuracy would increase as well, but diminishing gains were witnessed. If taking unrealistic values (<4 or >4096), in terms of our machines' performance, either the accuracy would be trampled or the expected time cost went from hours to days.

4.2.2 VAN - Batch Sizes

The default batch size for VAN was 128, which our GPUs was unable to contain and had to be reduced. A variety of batch sizes were tested (1, 2, 4, 8 etc.). In terms of final resulting WER and

CER, we determined that any batch size that is greater than 4 and less than the largest acceptable amount our machines would be reasonable and not deviate the accuracy and loss to exceed statistical standards.

4.2.3 VAN - Learning rate

The default learning rate set for Adam was 0.0001, which compared with our experimental values, would be the most reasonable value since all higher or lower values usually generates worse accuracy.

4.2.4 VAN - Number of Epochs

The default value for number of epochs was 5000, which has to be reduced significantly since our GPUs were not able to contain such amount of iterations. Within our conjecture, this parameter is what caused our results to differ the most from the originals. From authors' graphs of loss [8], we could discover that loss will enter curvature dominated regime before around 1000 epochs and drops greatly afterwards if pretraining and early stop features were not present; however, even with these features included, it still requires at least 50 epochs to reach a considerable decrease in loss. Unfortunately, our GPUs, at maximum, could only contain around 20 epochs.

4.2.5 TrOCR

The default values for hyper-parameters in TrOCR model were already fine tuned and any of our attempts did not result better accuracy, thus, we determined that there would be no necessity for experimenting further and potentially worsening its performance. The advantage of TrOCR model is that, unlike HMM models [1,2], it does not require a lot of pre-processing of input images.

4.3 Results Comparison Table

Architecture	CER% validation	WER% validation	CER% test	WER% test
Authors' line-level	4.49	18.22	4.25	17.14
Authors' VAN on lines	4.42	18.17	4.10	16.29
Authors' VAN on paragraphs	4.01	15.47	3.83	13.94
Ours line-level	4.43	16.77	4.03	16.54
Ours VAN on lines	4.12	15.86	4.11	14.33
Ours VAN on paragraphs	8.45	31.89	8.23	28.62
Author's TrOCR BASE	/	/	3.42	/
Author's TrOCR LARGE	/	/	2.89	/
Ours TrOCR BASE	/	/	3.40	/
Ours TrOCR LARGE	/	/	2.93	/

4.4 Custom Recognition Examples

All images are listed in the Appendix as Figure3-7, as we can see, the models usually perform accurately, and only perform inaccurately when the handwriting is intentionally messy or not highly recognizable even for human readers.

5 Conclusion

As we can derive from the comparison table and examples, these two models both have state-of-art accuracy when recognizing handwritten characters, surprisingly high accuracy even when the handwriting is a little bit messy. However, their performances will be limited or even worsen if not enough computation power is present.

References

[1] A. Kundu, Yang He and Mou-Yen Chen, "Alternatives to variable duration HMM in handwriting recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pp. 1275-1280, Nov. 1998, doi: 10.1109/34.730561.

[2] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya and F. Zamora-Martinez, "Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 4, pp. 767-779, April 2011, doi: 10.1109/TPAMI.2010.141.

[3] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, pp. 855-868, May 2009, doi: 10.1109/TPAMI.2008.137.

[4] J. Puigcerver, "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, pp. 67-72, doi: 10.1109/ICDAR.2017.20.

[5] Chakraborty, Chandralika Talukdar, P.H.. (2016). Issues and Limitations of HMM in Speech Processing: A Survey. International Journal of Computer Applications. 141. 13-17. 10.5120/ijca2016909693.

[6] H. Bunke, M. Roth, E.G. Schukat-Talamazzini, "Off-line cursive handwriting recognition using hidden markov models, Pattern Recognition", Volume 28, Issue 9, 1995, Pages 1399-1413, ISSN 0031-3203, https://doi.org/10.1016/0031-3203(95)00013-P.

[7] Hernandez Diaz, D., Qin, S., Ingle, R., Fujii, Y., and Bissacco, A., "Rethinking Text Line Recognition Models", <i>arXiv e-prints</i>, 2021.

[8] D. Coquenet, C. Chatelain and T. Paquet, "End-to-end Handwritten Paragraph Text Recognition Using a Vertical Attention Network", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1-1, 2022. Available: 10.1109/tpami.2022.3144899.

[9] Atienza, Rowel. (2021). Vision Transformer for Fast and Efficient Scene Text Recognition. 10.1007/978-3-030-86549-$8_2$1.

[10] Christoph Wick, Jochen Zöllner, and Tobias Grüning. 2021. Transformer for Handwritten Text Recognition Using Bidirectional Post-decoding. In Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part III. Springer-Verlag, Berlin, Heidelberg, 112–126. DOI:https://doi.org/10.1007/978-3-030-86334-0_8

[11] Li, M., "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models", <i>arXiv e-prints</i>, 2021.

Appendices

A Appendix: Figures

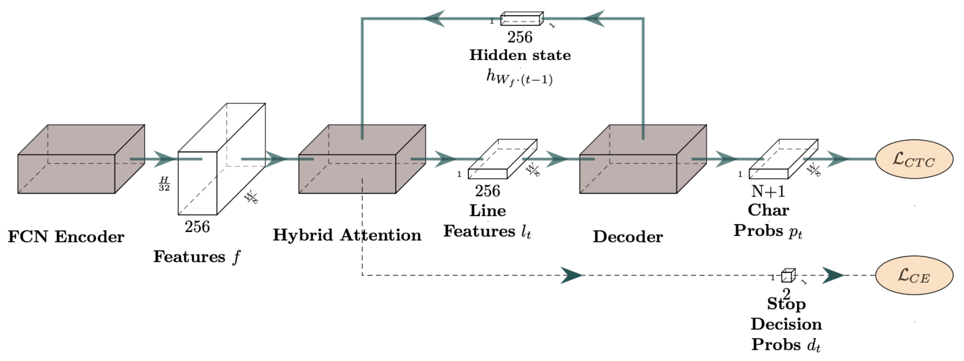


Figure 1: Model Architecture of VAN.[8]

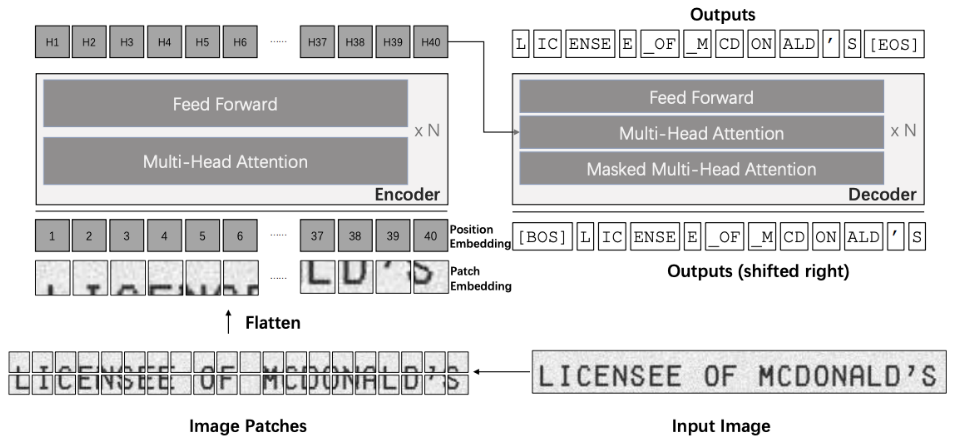
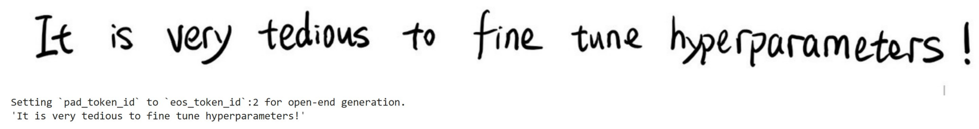
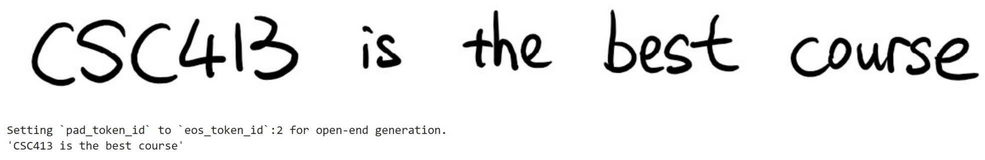


Figure 2: Model Architecture of TrOCR.[11]


It is very tedious to fine tune hyperparameters!

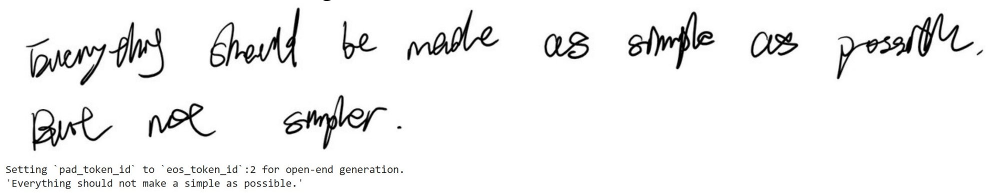
```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
'It is very tedious to fine tune hyperparameters!'
```

Figure 3: It is very tedious to fine tune hyperparameters.


CSC413 is the best course

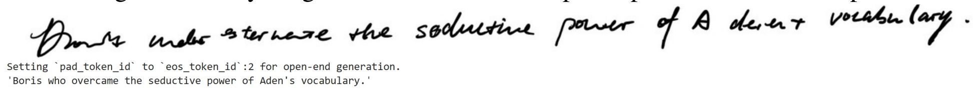
```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
'CSC413 is the best course'
```

Figure 4: CSC413 is the best course.


Everything should be made as simple as possible.
But not simpler.

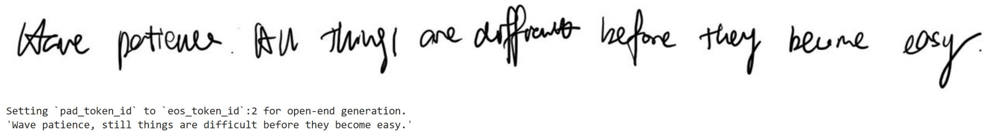
```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
'Everything should not make a simple as possible.'
```

Figure 5: Everything should be made as simple as possible, but not simpler.


Don't underestimate the seductive power of A decent vocabulary.

```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
'Boris who overcame the seductive power of Aden's vocabulary.'
```

Figure 6: Don't underestimate the seductive power of a decent vocabulary.


Have patience. All things are difficult before they become easy.

```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
'Wave patience, still things are difficult before they become easy.'
```

Figure 7: Have patience. All things are difficult before they become easy.

B Appendix: Code Base

All code can be found in: https://github.com/Specter43/OCR-for-handwriting-recognition.git