Connectionist Temporal Classification for Offline Handwritten Text Recognition

Oliver Nina[*]
University of Central Florida
Orlando, FL

ABSTRACT

Handwritten text recognition is an important problem that has many applications such as automatic indexing and transcription of historical documents.

In the last decade, there have been significant improvements with novel methods such as the Connectionist Temporal Classification (CTC) approach which combines generative Hidden Markov Models (HMM) based methods and recurrent neural networks to solve this problem.

CTC is currently the state of the art approach for offline handwriting recognition. Despite such improvements, little work has been done on speeding up the training phase of the neural networks and parallelizing the pipeline using GPUs for this particular task. Furthermore, there is also little work on open source libraries to perform further research in this area.

In this paper we give an overview of the CTC algorithm as well as discuss how such algorithm could be speed up using a parallelized implementation.

1. INTRODUCTION

Handwritten text recognition (HTR) is an open field of research and a relevant problem that helps automatically process historical documents.

In recent years great advances in deep learning and computer vision have allowed improvements on document and image processing including HTR.

Currently the state of the art on HTR is the Connectionist Temporal Classification algorithm which was used to win a recent HTR competition [7] on the transScriptorium dataset [8].

Despite such advances in this field, little has been done to produce open source projects that address this problem as well as methods that utilize graphical process units (GPUs) to speed up the training phase.

[*]corresponding author's email address onina@eecs.ucf.edu

In this paper we give an brief overview of the CTC algorithm for recognizing handwritten text images and discuss possible solutions to improve performance and accuracy.

2. BACKGROUND

Handwritten text recognition (HTR) is a problem that has been studied for many years.

Some of the earlier methods on handwriting recognition utilized Hidden Markov Models (HMM) such as [3] others used neural networks such as in [4] and other methods used a combination of both [9]

Newer methods involve more powerful models using recurrent neural networks as in Connectionist Temporal Classification [4] which is currently the state of the art in handwriting recognition.

The CTC algorithm has been used on commercial and open source software such as in ocropy [1]. Although there are some open source projects that provide libraries for CTC such as ocropy [1] or tesserac [2], however, such tools are currently only used for typed text or OCR and not for handwritten text.

In this paper we describe the use of the ocrpy library for handwriting recognition and discuss CTC algorithm which is behind the learning procedure of such library. We also discuss solutions to improve training time.

3. PROPOSED METHOD

3.1 Normalization

The first step of the process is the proper normalization of the text. In our method we assume that lines of the document have been segmented accordingly and other operations such as rotation of the pages have been performed already.

3.1.1 De-slant

Given an image of a line segment we proceed to calculate the slant angle needed to remove the text slant from the text. We examine training samples to approximate a constant value as the angle to perform a shear operation in an affine transformation. We see the results of our de-slant approach in Figure 1b

3.1.2 Contrast Normalization

We also perform contrast normalization on the values of the image by subtracting the pixel values from the highest pixel value in the image which will cause the histogram of the image to stretch through the whole gamma of gray scale values. We can see the result of our contrast normalization in Figure 1c

3.1.3 *RNN Input Normalization*

Finally in order to input the image into our Recurrent neural network which expects a set number of input pixels at every time step, we normalize the size of the height of the image to fit to the RNN input size which is 48. This size normalization of the image could cause loss on the quality of the image. Higher quality of the images could be used for RNNs with higher number of parameters, however, it could affect the speed of the training phase.

3.1.4 *Feature Extraction*

Each feature for the temporal space is obtained at every pixel column of the image. Hence, several columns of the image will have the same label which will correspond to one character in the image.

3.2 Long Short Term Memory

In this part of our algorithm we employ a recurrent neural network known as Long Short Term Memory (LSTM) [5, 6]. LSTMs have been used successfully for offline recognition.

Traditional LSTMs, also known as *vanilla* LSTMs, differ from regular RNNs by adding gating functions that allows them to memorize or forget inputs seen in previous time steps.

The main gating functions defined for the LSTM formulations are: an input gate i_t which prevents the cell unit to be switched off by irrelevant inputs, a forget gate f_t which diminishes the effects of previous cell states, an update gate z_t which updates the state of the cell based on the input gate, a memory cell unit c_t which stores the state at the current time step, and an output gate o_t which reduces perturbation of the cell output to other consecutive cells.

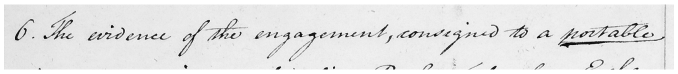
The basic formulation of LSTM is defined as follows:

$$\begin{aligned} \hat{y}_t &= f\left(s_t\right) \\ s_t &= W^{(y)} h_t \\ i_t &= \sigma(W^{(i)} x_t + U^{(i)} h_{t-1}) \\ f_t &= \sigma(W^{(f)} x_t + U^{(f)} h_{t-1}) \\ o_t &= \sigma(W^{(o)} x_t + U^{(o)} h_{t-1}) \\ z_t &= g(W^{(z)} x_t + U^{(z)} h_{t-1}) \\ c_t &= i_t \circ z_t + f_t \circ c_{t-1} \\ h_t &= o_t \circ g(c_t) \end{aligned} \tag{1}$$

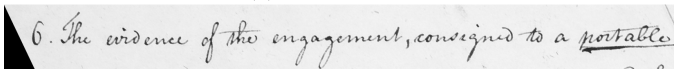
In a regular LSTM network f is a softmax function that is ouput for classification purposes, however, in our application f is the input to a CTC layer which will be explained later in this paper, σ represents the sigmoid function $\sigma(a) = \frac{1}{1+e^{-a}}$ and g is the hyperbolic tangent function $g(a) = \frac{1+e^{-2a}}{1-e^{-2a}}$. The operation $\circ$ represents the Hadamard product. $W^{(\star)}$ and $U^{(\star)}$, where $\star = \{i, f, o, z\}$, are the weights being learned.

Note that for simplicity, we are omitting the bias terms for i_t, f_t, o_t, z_t. Because c_t is a linear function composed of the sum of two products, such linearity helps lessen the effects of the vanishing gradient problem during backpropagation as well as modulate the influence of current inputs and previous cell states.

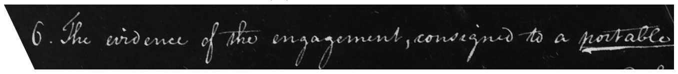
3.3 Connectionist Temporal Classification



(a) Original Image



(b) De-slanting



(c) Contrast Normalization

Figure 1: Line Image Normalization Process

The connectionist temporal classification(CTC) [4] is an output layer used for optical character recognition and uses as inputs a normalized softmax classification layer which is output by the LSTM network. This normalized output is used to approximate the conditional probabilities of observing a character or a blank k at time t. The conditional probability $p(\pi|x)$ of a specific path of consecutive characters could be obtained by the sums of all the paths that could have the characters in them. This is achieve by using the forward-backward algorithm to obtain the total probability. More specifically:

$$p(l|x) = \sum_{\pi \epsilon B^{-1}(l)} p(\pi|x) \tag{2}$$

We allow all the paths to start with either a blank (b) or a label l. Hence, the label sequence probability is given by:

$$p(l|x) = \sum_{s=1}^{l'} \alpha_s \beta_s \tag{3}$$

where α corresponds to the forward variables and β is the backward variables.

The objective function of CTC is given by:

$$O = -\sum_{(x,z)\epsilon S} \log p(z|x) \tag{4}$$

Which we can differentiate with respect to the parameters of the network and train through gradient descent. For the differentiation details of the CTC algorithm we refer the reader to [4]. By using CTC in this way allows us to perform recognition of the text at every time step t without having to previously segment the characters individually from the image.

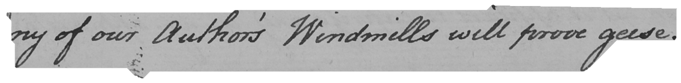
3.3.1 *CTC decoding*

At every time step after training we can perform a classification of every time step by picking the label with the highest probability In this way:

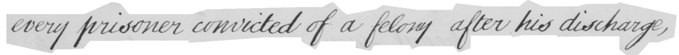
$$l' = \arg\max_{l} p(l|x) \tag{5}$$

3.4 Transfer Learning

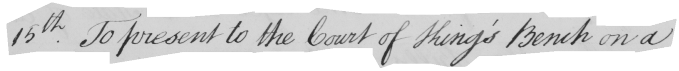
In order to speed up the training phase for our CPU implementation, we use the weights of a model trained on typed


ny of our Author's Windmills will prove geese.

(a) TRU: “ny of our Author ’ s Windmills will prove geese .”
ALN:”ny of f our Author ’ s Windmils wil prove geee .”
OUT:’teteee thetee theeeette ant taene faee’


every prisoner convicted of a felony after his discharge,

(b) TRU: ”every prisoner convicted of a felony after his discharge ,”
ALN: ”evvery prisoner conviceed of a felony after his dischage ,”
OUT: ”teef teeeeeee sasestet of o tfeey ofostee teeteen”


15th. To present to the Court of King's Bench on a

(c) TRU: ”15th . To present to the Court of King ’ s Bench on a”
ALN: ”15thh . To presnt to the Court of King ’ s Bench on n a”
OUT: ’aof sahaeante the fantf theg theene o ne’

Figure 2: Output of our training phase on a CPU after 157K iterations (14 epochs)

written text for 100K iterations. Such model is used to initialize the weights of a new neural network trained on hand written text. Such initialization, helps improve the training time by allowing to fine tune the weights for hand written text.

4. RESULTS

Data: We use the Transcriptorium dataset [8] to train our neural network. We use the first batch of line images for training that contains more than 11K images. Figure 2 shows some of the training images of this dataset. Figure 2 shows also the result of our model trained on a CPU version of ocropy after 157K iterations. TRU refers to ground truth, ALN is the output of CTC aligned with the training text and OUT is the final output of the network.

Figures 3a, 3b, 3c, show that after about 14 epochs the network start to align its predictions very close to the original text and on that image we can also see that the ouput of the network starts to take some form of the actual text.

In order to train our network on a CPU for more than 10 epochs as seen in Figure 2, we needed about two days for the training process to run.

If we increase the number of iterations to 465K, we can see that our model starts learning some of the words that has frequently seen in the training set as we can see in Figure 3.

5. IMPLEMENTATION DETAILS

We used the implementation of the LSTM in Ocrpy. However, some of the issues with Ocrpy is that the training phase is computationally expensive requiring several days to train a simple model.

Another drawback of Ocrpy is that to perform experiments with other types of recurrent units, it requires to manually calculate the derivation of the objective function.

Hence, one way to alleviate both problems is by using Theano. We experimented with a new version of the CTC algorithm based in Theano which is currently under development. The Theano version of the CTC algorithm allows us to compute the gradient descent computation on the GPU. It also helps us experiment with other types of recurrent units more easily

Our code for our cpu version of our HTR system is an open source project on Github at:

https://github.com/olivernina/ocropy.

The code for our gpu version which is under development is found at:

https://github.com/olivernina/htr-ctc

6. CONCLUSIONS

In this paper we present an overview of the CTC algorithm for offline handwriting recognition. We also discuss alternatives to improve training performance of the CTC algorithm and research with different recurrent units.

7. REFERENCES

[1] Ocropy. `https://github.com/tmbdev/ocropy`.

[2] Tesseract. `https://github.com/tesseract-ocr/tesseract`.

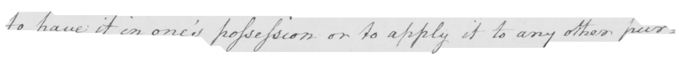
[3] A. El-Yacoubi, R. Sabourin, M. Gilloux, C. Suen, et al. Off-line handwritten word recognition using hidden markov models.

[4] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

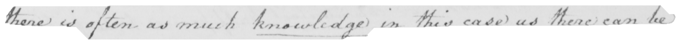
[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] H. Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

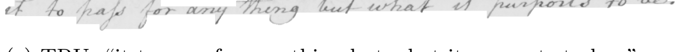
[7] J. Sanchez, A. Toselli, V. Romero, and E. Vidal. Icdar 2015 competition htrts: Handwritten text recognition on the transcriptorium dataset. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1166–1170, Aug 2015.


to have it in one's poſseſsion or to apply it to any other pur=

(a) TRU: “to have it in one ’ s possession or to apply it to any other pur=”
ALN: “to have it in one ’s possession or to apply it to any other puur=”
OUT: “tf bane it in oneis posseseon or to apply it to any other puu”


there is often as much knowledge in this case as there can be

(b) TRU: “there is often as much knowledge in this case as there can be”
ALN: “there is often as much knowledge in this case as there can be”
OUT: “hore is often as muct tnnledge in this case as there can te”


it to paſs for any thing but what it purports to be.

(c) TRU: “it to pass for any thing but what it purports to be .”
ALN: “it to pass for any thing but what it purports to be .”
OUT: “tt to pats tor any thing but what it purports to t”

Figure 3: Output of our training phase on a CPU after 468K iterations (40 epochs)

[8] J. A. Sánchez, G. Mühlberger, B. Gatos, P. Schofield, K. Depuydt, R. M. Davis, E. Vidal, and J. de Does. transcriptorium: a european project on handwritten text recognition. In *Proceedings of the 2013 ACM symposium on Document engineering*, pages 227–228. ACM, 2013.

[9] Y. Tay, P.-M. Lallican, M. Khalid, C. Viard-Gaudin, and S. Knerr. Offline handwritten word recognition using a hybrid neural network and hidden markov model. In *Signal Processing and its Applications, Sixth International, Symposium on. 2001*, volume 2, pages 382–385 vol.2, 2001.