Data Mining Term Report

Student：**Chung-Chia Cheng**（鄭中嘉）
Student ID：**L46104020**
Space Weather Lab, Department of Earth Science, National Cheng Kung University

Table of the report

The term report is structured as follows, in **section I. Brief Introduction**, the information about the author and the paper will be listed; in **section II. Motivation and Contribution**, the background of this method and its potential application will be shortly shown; in **section III. Database Description**, the outline of the IAM database (containing handwritten english text) will be introduced; in **section IV. Methodology**, we will go through several methods involved in this task. The methods discussed in this section will not be limited to deep learning algorithms. Other needed strategies will be introduced as well. In **section V. Results**, some of the results from the original paper will be displayed. In **section VI. Discussion**, we will focus on two questions which are related to the results presented in the original paper. I will cite all sources of information used in this report in **section VII. Reference**, and finally, the basics of deep learning techniques will be introduced in **section VIII. Appendix**.

I. Brief Introduction

I chose the paper published in May 2020 in the journal , “International Research Journal of Engineering and Technology (IRJET)”, titled “Handwritten Text Classification using Deep Learning”. The following table (**Table 1)** shows the information about the original paper **(Vidushi, 2020)**.

Table 1. Information about the original paper

Paper	Handwritten Text Classification using Deep Learning
Author	Vidushi Garg
From	Department of Information Technology, Maharaja Agrasen Institute of Technology, New Delhi, India

II. Motivation

This paper aims to propose a method to **transcribe handwritten text into digital text**, as shown in **Figure 1**. Because it's difficult to store and access physical data (i.e., the texts we've written down) with efficiency. Manual labor is required in order to maintain proper organization of the data.

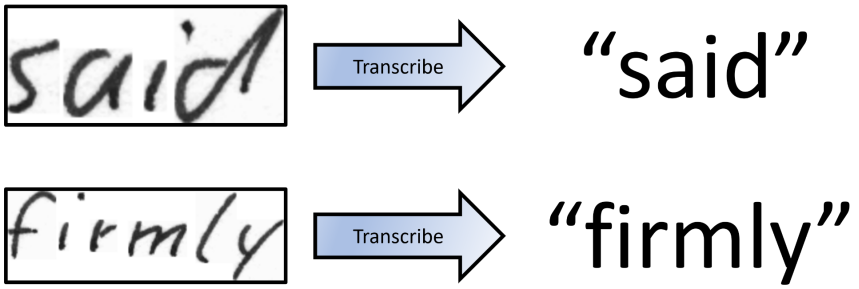

Figure 1. Transcribe handwritten text into digital text.

Besides for the maintenance purpose, this technique can also be used in many aspects in our daily life. For example, by using this technique, we can speed up the workflow for operators who are responsible for keying information into the computer. The diagram of the possible workflow is shown in **Figure 2**. Instead of typing every handwritten text into a computer, it's possible to use the method proposed in the paper first, and the operators check the results afterwards. Not only does the workload decrease, the efficiency might also increase.

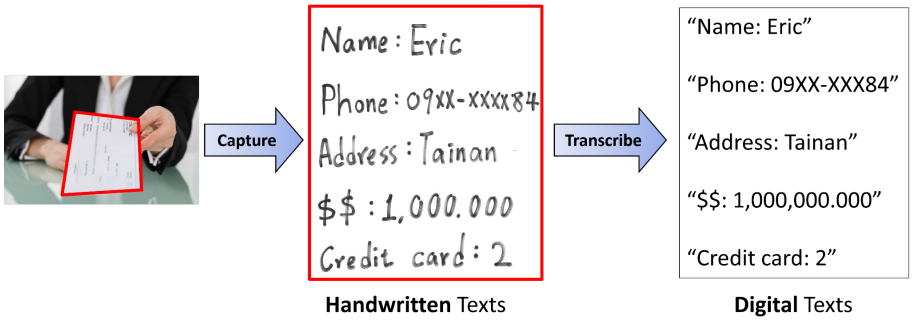

Figure 2. Possible workflow

III. Database Description

The author took the IAM dataset as training and validation datasets. The IAM Handwriting Database contains forms of handwritten English text, which were scanned at a resolution of 300dpi and saved as PNG images with 256 gray levels.

According to the **official website of IAM Handwriting Database**, the database is updated to version 3.0 and structured as follows, and some of the examples are shown in **Figure 3**.

- 657 writers contributed samples of their handwriting
- 1,539 pages of scanned text
- 5,685 isolated and labeled sentences
- 13,353 isolated and labeled text lines
- 115,320 isolated and labeled words.

(Visit https://fki.tic.heia-fr.ch/databases/iam-handwriting-database for more information.)

Note that, the words have been extracted from pages of scanned text using automatic segmentation scheme and were verified manually. The segmentation scheme involves transcriptions of the text lines and a hidden Markov model (HMM) based recognition system (c.f., M and H, 2002).

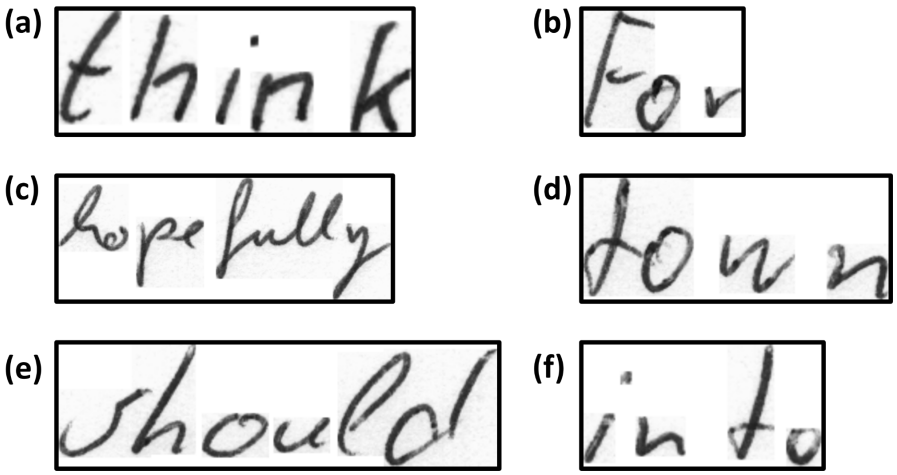

Figure 3. Samples from the IAM Database
(a) sample word of “think”, (b) sample word of “For”, (c) sample word of “hopefully”, (d) sample word of “town”, (e) sample word of “should”, and (f) sample word of “into”

IV. Methodology

With the handwritten text dataset available, there are two steps involved before using deep learning algorithms, which are **data preprocessing** and **data augmentation**. Following these two steps is an end-to-end deep learning model which could be divided into 3 parts, including **Convolutional Neural Network**, **Long Short-Term Memory Neural Network** and **Connectionist Temporal Classification Decoder**. In this section, we will dive into each method mentioned above and explain their mathematical concepts.

A. Data Preprocessing

The input size of the images for the Convolutional Neural Network is set to 128 x 32. However, the samples from the IAM database do not have exactly the size, therefore, resizing the sample images until it has the width of 128 or a height of 32 is needed. After resizing without distortion, the samples might still not have the size of 128 x 32, in this case, the empty part will be padded with white color to make sure samples have the size of 128 x 32. **Figure 4** shows how resizing and padding were used.

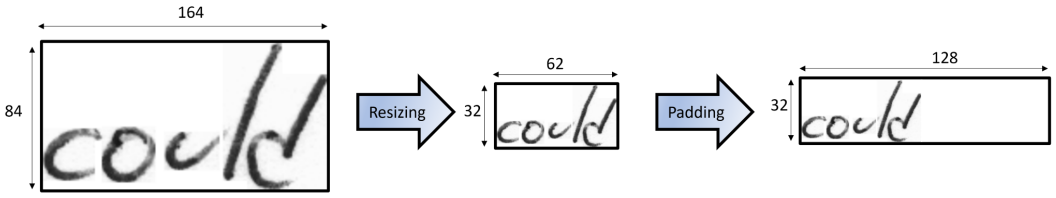

Figure 4. Resizing and Padding on the sample from the IAM Database

After observing and analyzing the samples from the IAM Database(see **Figure 3**), the author points out that there are three noticeable patterns. Firstly, images have high contrast; secondly, words are tightly cropped; and thirdly, words are in bold writing style. Hence, when trying to use the trained handwritten recognition model, users should first make sure that the input image meets these three conditions (i.e., high contrast, tightly cropped, and bold writing style).

B. Data Augmentation

The author mentioned that data augmentation is a strategy to significantly increase the diversity of data available for the training models without actually collecting data. The commonly used techniques are cropping, padding, and horizontal flipping.

However, specifically for the task of handwritten text recognition, I have different opinions that if applying **cropping**, the edge of the words might be cropped out; if using **padding**, the samples from the IAM Database will lose their original characteristics (tightly cropped); and if applying **horizontal flipping**, the order inside the words will be reversed. Therefore, although the author mentioned the benefits of data augmentation, I still think in this scenario, at least the three methods mentioned by the author should **NOT** be used (see **Figure 5**). Instead, I recommend using randomly resizing without keeping the aspect ratio as an approach to achieve the purpose of data augmentation.

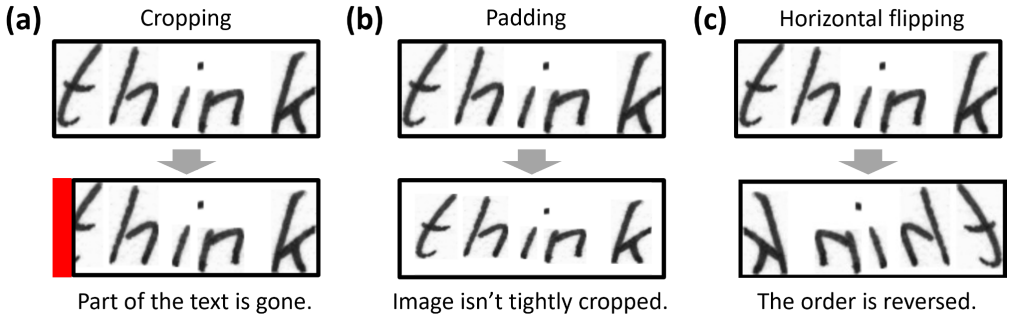

Figure 5. The risk of applying cropping, padding, and horizontal flipping (a) cropping, (b) padding, and (c) horizontal flipping

From **part C** to **part E**, we will dive into an end-to-end trainable Artificial Neural Network (ANN). This ANN contains 5 convolutional layers and 2 layers of bidirectional LSTM units with a hidden size of 256. Followed by the Bidirectional LSTM layer is the CTC layer. The CTC layer is responsible for transcribing a sequence of probability distributions into the predicted sequence (result). The overall structure of the ANN is shown in **Figure 6**, also, there is a brief introduction of how to train an end-to-end deep neural network in **section VIII. Appendix A**.

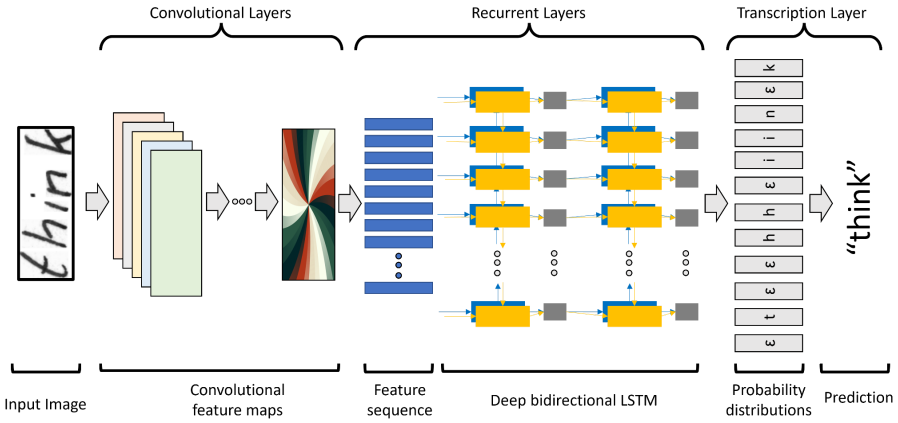

Figure 6. Overall structure of the ANN

C. Convolutional Neural Network

Convolutional neural network (CNN) is used to extract useful information from the input image with size of 128 x 32. The extracted information is called a feature map, which is in the form of a three dimensional matrix, and (is believed) contains semantic information that helps for the classification task.

A CNN consists of an input and output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that *convolve* (*see **section VIII. Appendix B***) with a multiplication or other dot product. The *activation function* (*see **section VIII. Appendix C***) is commonly a RELU layer, and is subsequently followed by additional convolutions such as *pooling* (*see **section VIII. Appendix D***) layers. **Figure 7** shows the detailed structure of CNN. One thing worth mentioning is that in each convolutional layer, the channel of the output feature map (the dimensions of a feature map is usually defined as width x height x channel) is equal to the number of kernels used in each convolutional layer. The trainable neurons inside the kernels will be optimized throughout the training phase.

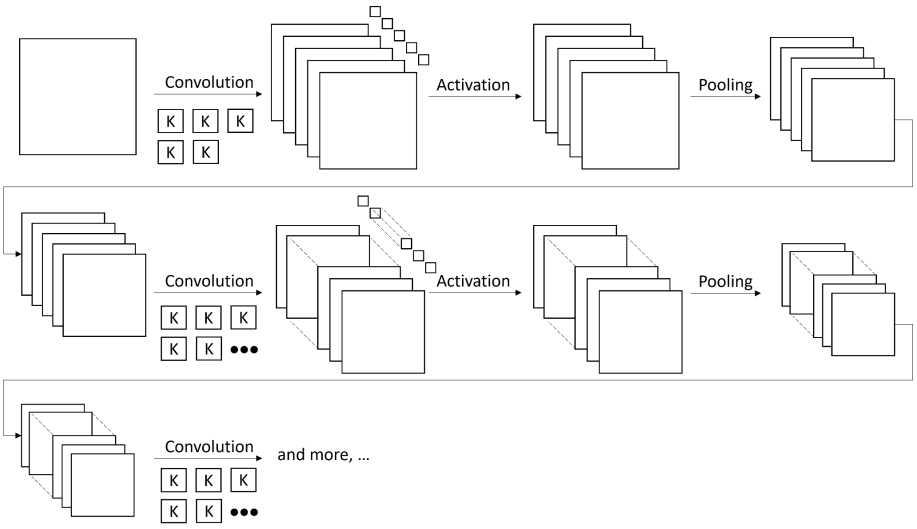

Figure 7. The structure of CNN

Once the CNN outputs the feature map, the feature map will be mapped into a sequence-like signal called "**feature sequence**" (as a video consists of a sequence of frames in each timestamp). The feature sequence is then passed to Long Short-Term Memory (LSTM) Neural Network (see **Figure 6**).

D. Long Short-Term Memory Neural Network

As a sequence model, LSTM is used to produce a **sequence of probability distributions** based on the **feature sequence** derived from CNN.

As mentioned earlier, in the structure of ANN presented, 2 layers of bidirectional LSTM units with hidden size of 256 are used. A single LSTM unit is composed of a memory cell, an input gate, an output gate, and a forget gate, where the memory cell remembers values over arbitrary time intervals and three gates regulate the flow of information into and out of the cell (see **Figure 8**). For details about LSTM, please refer to the published paper (c.f., Sak et al., 2014).

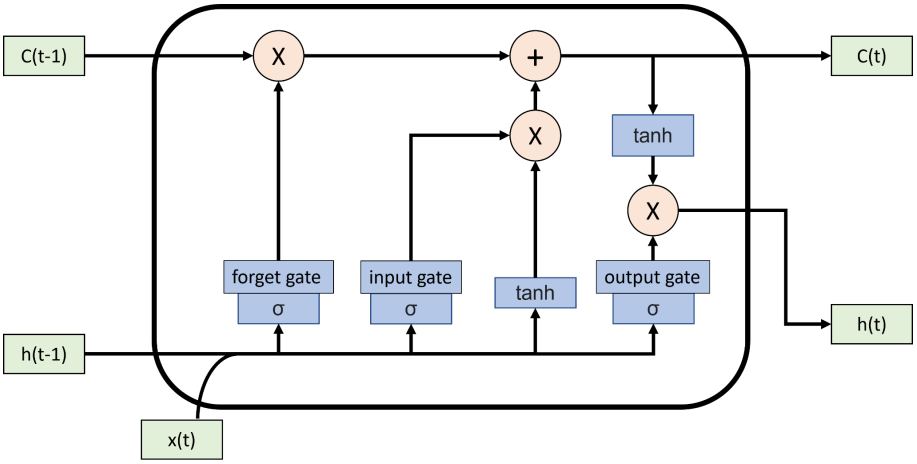

Figure 8. Architecture of LSTM unit

LSTM networks are well-suited to classifying, processing and making predictions based on time series data. Also, LSTM is developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional recurrent neural networks. With the usage of bidirectional LSTM, the information will be passed not just from left to right, but also from right to left (see **Figure 6**), so, before it outputs the **sequence of probability distributions**, it gets to see the information of the entire input sequence. It's why bidirectional LSTM usually has better performance than pure LSTM.

Finally, after finishing the part of LSTM, the **sequence of probability distributions** will then be passed to Connectionist Temporal Classification (CTC) Decoder.

E. Connectionist Temporal Classification Decoder

In the training phase, CTC Decoder serves to calculate the loss and propagate the signal of the loss back to the LSTM neural network and CNN; as for phase of inference, the CTC outputs the predicted texts according to the sequence of probability distributions derived from LSTM Neural Network.

There are two reasons that the author used CTC as the tool to transcript probability distributions into texts. Firstly, there is ambiguity (no clear boundaries) in the alignment of the input image and the output text (see **Figure 9**), for which, CTC can properly handle. Secondly, CTC is good at sequence-to-sequence problems. In the following, I will explain the mathematical concepts in CTC.

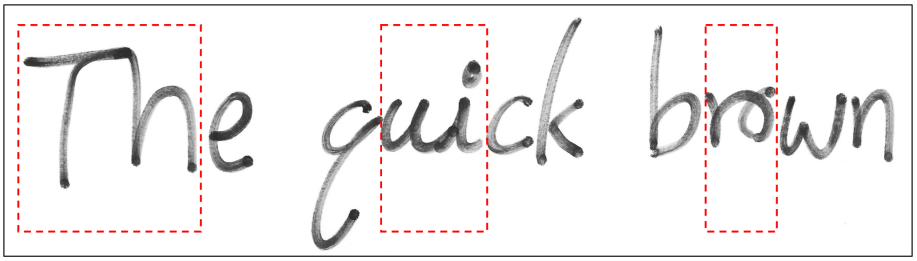

Figure 9. Texts can not be clearly separated

Given input X, CTC will calculate $P(Y|X)$ for all possible Y. With $P(Y|X)$, we can predict the most likely output of texts. Before we dive in more, let's see the simplest way to align the input and the output. Please refer to **Figure 10**, we provide X in each timestamp a possible Y. However, by using this method to align input, we will never have an output with consecutive texts. For example, consider the input image of "apple", which will lead us the output of "aple".

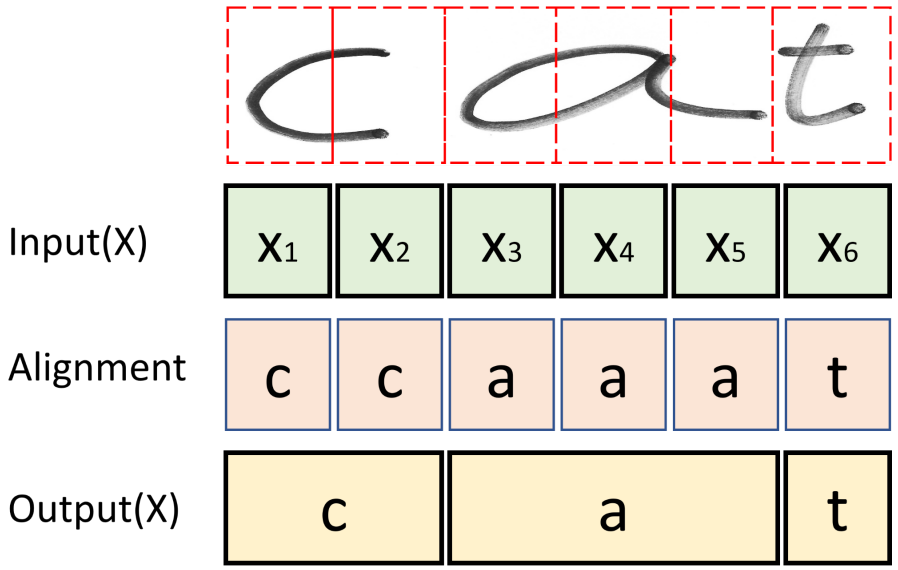

Figure 10. Simple alignment

In order to overcome the problem mentioned above, CTC introduced a new symbol, ε, called blank token. By using it, we can use it to distinguish consecutive texts (see **Figure 11**).

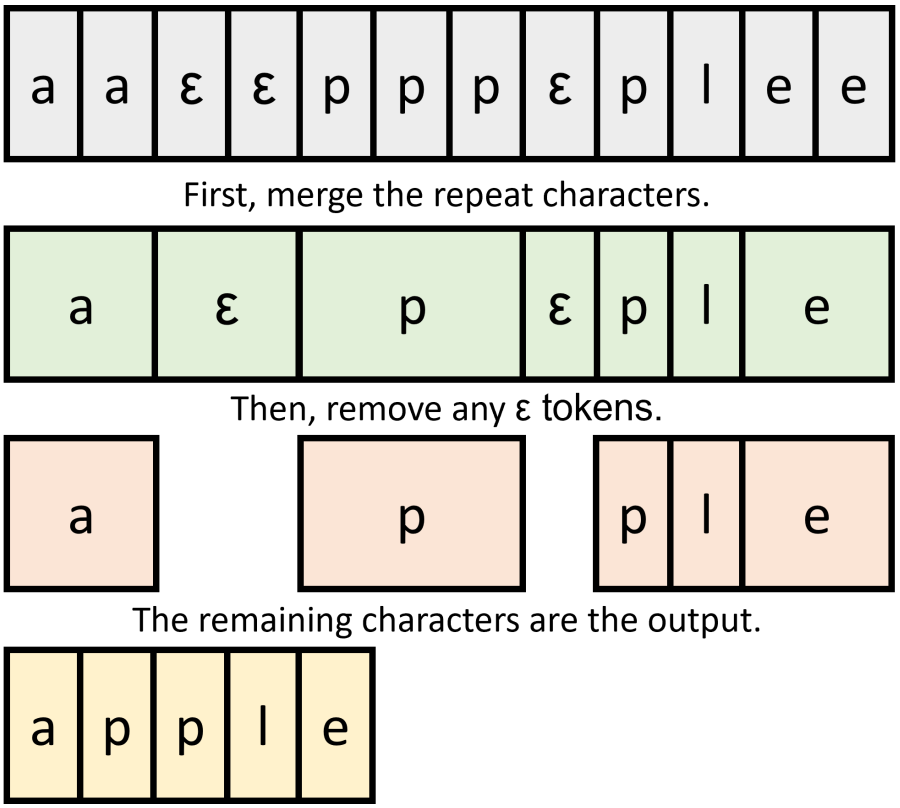

Figure 11. CTC alignment

Now, let's combine the output of LSTM (i.e., a sequence of probability distributions) and the idea of CTC alignment. Assume at timestamp t. The probability distribution is defined as $p_t(a|X)$, where a means a single text which belongs to $S = \{a, b, c, ..., x, y, z, \varepsilon\}$. With sequence length of T, in theory, there will be 27^T possible ways to align the input to the

output. If we assume the conditional probabilities are conditionally independent, then formally, with a pair of (X, Y), we can formulate the situation as **Formula 1.**

$$P(Y|X) = \prod_{t=1}^{T} P_t(a_t|x) \qquad \text{(Formula 1)}$$

During training phase, our purpose is to maximize $P(Y|X)$. Therefore, in terms of loss function, we can formulate the CTC loss function as **Formula 2**.

$$L = \sum_{(X,Y)\in D} -\log P(Y|X)\,, where\ D\ is\ Batch \qquad \text{(Formula 2)}$$

Guided by this loss function, ANN will adjust its trainable parameters during training phase in order to find Y^* which can be represented as **Formula 3**.

$$Y^* = \operatorname{argmax} P(Y|X) \qquad \text{(Formula 3)}$$

If we would like to find the exact answer, then we have to traverse all possible ways of alignment, the cost we need to afford is in the order of 27^T in our case. There are two common algorithms people use to find the approximated answer, the first one is **greedy search**; and the second one is **beam search**. By using **greedy search**, in each timestamp, the output with the largest possibility will be chosen. But in some cases (see **Figure 12**), **greedy search** might output the wrong answer, so, **beam search** would be used to compensate for this situation. **Beam search** will choose several outputs with largest possibilities as candidates in each time stamp until it finishes searching.

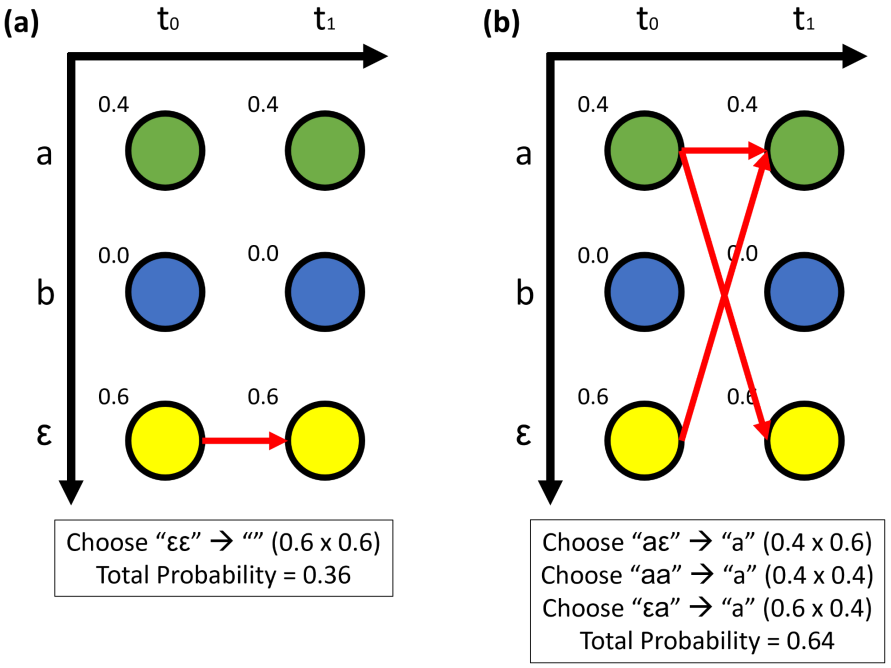

Figure 12. Situation that greedy search might be wrong (Assume the answer is "a")
(a) greedy search and (b) beam search

V. Results

This section shows the results of the original paper (Vidushi, 2020), including the screenshots of training, validation on the IAM Database and testing on unseen images (see **Figure 13**).

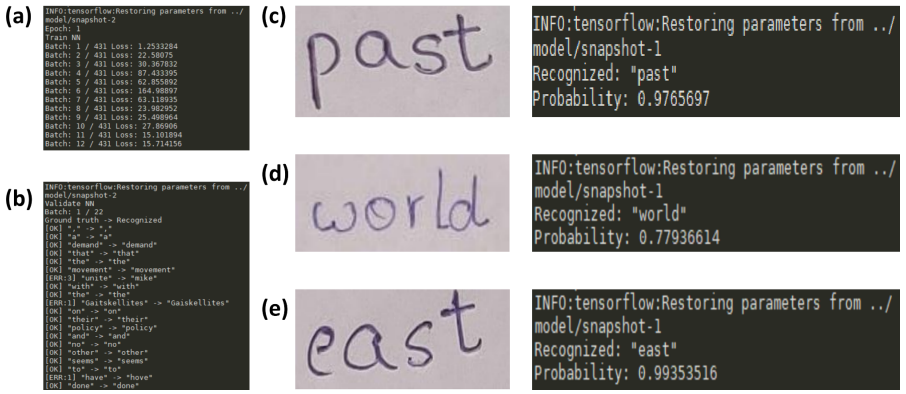

Figure 13. Results of the original paper
(a) training, (b) validation, and (c)(d)(e) testing on unseen samples

VI. Discussion

In this section, I will answer two questions. The first one is whether the results of the paper are obsolete, and the second one is whether it is possible to improve the results. Also, a short conclusion will be stated as well.

A. Whether the results of the paper are obsolete

In the aspect of **transcribing handwritten text into digital text**, the entire architecture of ANN is called Convolutional Recurrent Neural Network (CRNN) + CTC. Because this architecture could be trained from end to end and it's suitable for sequence-to-sequence problems, it's still a chosen architecture for people when they are facing similar problems (where the input is an image, and the output is the digital text). For example, **license plate detection** and **gesture recognition** could be solved by applying CRNN + CTC.

B. Whether it is possible to improve the results

The answer is yes. Although in **part C**, I said the architecture itself is still on the main trend. However, with the rapid development in the world of deep learning, there are some new techniques that could be integrated into the structure of CRNN.

Before jumping to the potential new techniques, we first review the author's opinions. In the report, ANN is composed of 5 layers of CNN, and 2 layers of bidirectional LSTM. The author mentioned that 7 layers of CNN can be used instead of 5 layers, with that, ANN might have greater ability to extract information from the input images. As for the type of LSTM, the author said, it might also be helpful to replace bidirectional LSTM by Multidimensional LSTM (MDLSTM) layer to also propagate information along the vertical image axis (Paul et al., 2016).

Besides that I think only randomly resizing without keeping aspect ratio (mentioned in **section IV. Methodology B**) is a more appropriate method for doing data augmentation, I have other two ideas that might help to increase the overall performance.

The first one is to modify the CTC loss function. Currently the loss function is defined as **formula 2**. The potential issue for that is ANN might tend to predict a shorter word because with more texts in a word, the lower the possibility it will be. Therefore, the loss function should be integrated by a language model as **formula 4**. With better performance in language model, and longer length of the word, the overall loss will be lower.

$$L = \sum -\log P(Y|X) + \frac{\alpha}{P(Y)} + \frac{\beta}{L(Y)}, \qquad \text{(Formula 4)}$$

$where\ P(Y)\ is\ a\ language\ model, and\ L(Y)\ is\ the\ length\ of\ the\ word$

The second one is to replace bidirectional LSTM by attention-based sequence-to-sequence models like Transformer. With an attention mechanism operating, it's more likely for the network to find the relationship among elements in every timestamp. Therefore, it may leave ANN with better performance.

C. Final words

In this report, we first understand the background of the problem setting (i.e., **transcribing handwritten text into digital text**). The source of datasets is introduced, and we've seen several samples from the IAM Database. There are 5 topics discussed in the section of Methodology, including data preprocessing, data augmentation, CNN, LSTM and CTC. Finally we saw the results derived from the author, and have more discussion in this section. For reference, please check **section VII. Reference**. As for topics like deep neural network, convolution, activation function, or pooling layer, please check **section VIII. Appendix**.

VII. Reference

M, Z., & H, B. (2002). Automatic segmentation of the IAM off-line database for handwritten English text. *Object recognition supported by user interaction for service robots*, *4*, 35-39.

Sak, H., Senior, A. W., & Beaufays, F. (2014). Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *ArXiv*, *abs/1402.1128*.

Paul, V., Patrick, D., & Hermann, N. (2016). Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks. *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 228-233. 10.1109/ICFHR.2016.0052

Vidushi. (2020). Handwritten Text Classification using Deep Learning. *International Research Journal of Engineering and Technology (IRJET)*, *7*(5), 3485 - 3490.

VIII. Appendix

A. Deep Neural Network

Deep neural networks have one input layer, one output layer, and several hidden layers. Based on the problem we are interested in, we can design the output layer in different forms. For example, if we are interested in a binary classification problem, then we want a output of a single value, so only one neuron is needed in this case; if we are interested in a multi-class classification problem, we want an output of a vector with length of 10, so 10 neurons will be needed in this case (see **Figure 14**).

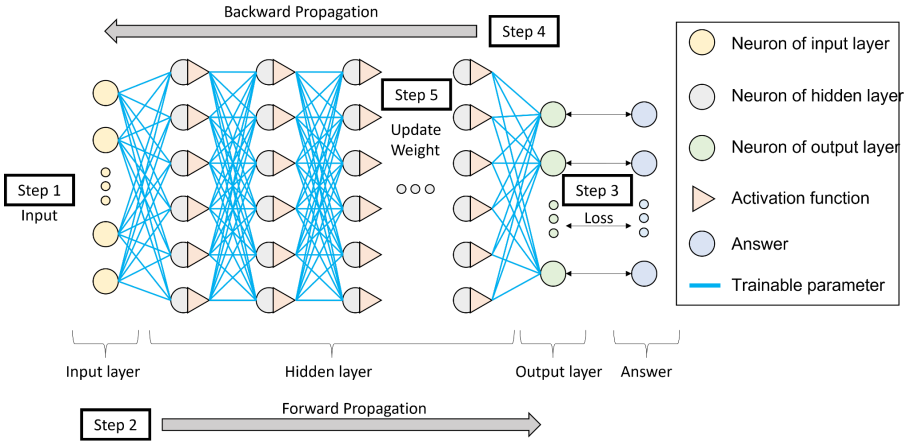

Figure 14. Deep neural network

When mentioning the training phase, there are 5 steps involved. In the first step, we need to pass in a batch of training instances into the input layer. In the second step, the neural network will perform forward propagation to pass the signal from left to right until it reaches the output layer. And, the loss will be calculated in the third step. In the fourth step, the signal of loss will be back propagated (from right to left) until reaching the input layer. Finally, in the last step, all the trainable parameters will be updated based on the signal of loss. By iteratively doing the mentioned 5 steps, ideally, the performance of the network will be better and better through the time.

Note that, the trainable parameters, often called weights, are the links between adjacent neurons whose values will be used (i.e., matrix multiplication) during forward propagation and will be adjusted during backward propagation.

B. Convolution

When the input is not a one-dimensional vector, but a two- or three-dimensional matrix, then often, the structure of the deep neural network will use convolutional neural networks. Different from the trainable parameters mentioned in **Part A**, the trainable parameters is not just a link between adjacent neurons, it becomes a two-dimensional kernel. We use the kernel to convolve on the input image, and we call the output result **feature map**. Similar to defining the dimension of an image as Width x Height x Channel (W x H x C), the dimension of a feature map is also W x H x C. The number of the channel of the feature map is the number of the kernels we used when convolving.

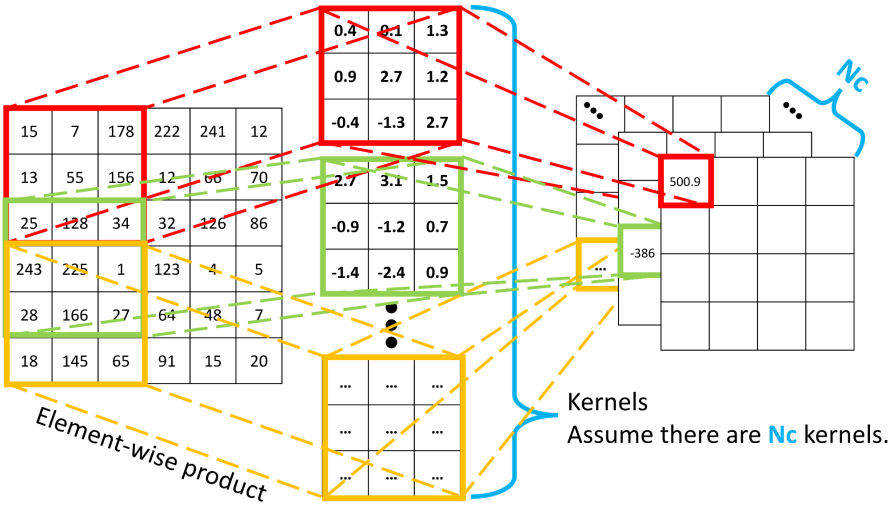

Figure 15. convolution

C. Activation Function

In order to empower the deep neural network to map information with nonlinearity, activation functions are usually introduced. They will be placed right after each neuron. Therefore, in a hidden layer, when performing forward propagation, the signal will be firstly passed through the trainable parameters, and then be passed through the activation functions afterwards.

There are three common activation functions (see **Figure 16**), including sigmoid whose output is ranging from 0 to 1, tanh whose output is ranging from -1 to 1, and relu whose output is equal to or greater than 0. In the paper, the author decided to use **relu**.

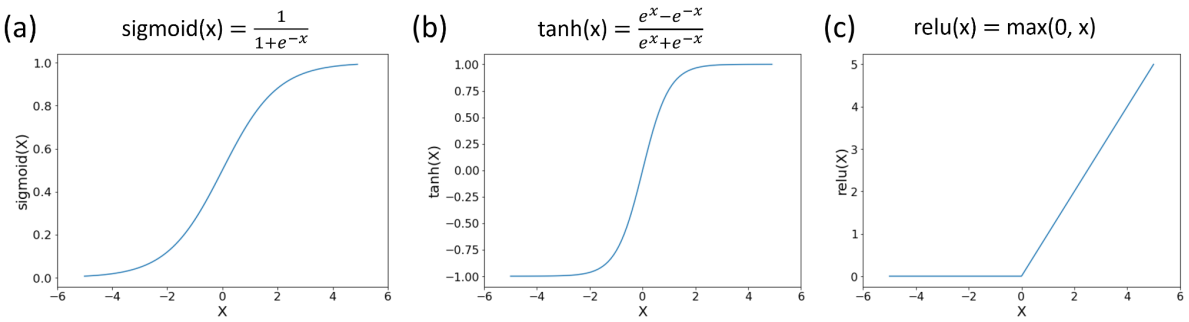

Figure 16. Activation functions
(a) sigmoid, (b) tanh, and (c) relu

D. Pooling Layer

When constructing CNN, besides using convolutional layers, pooling layers are also used to generate a summary statistic for small regions. The common pooling layers are **average pooling layer** and **max pooling layer** (see **Figure 17**). Average pooling will calculate the average value for a specific region; as for max pooling, it will find the maximum value in that specific region. The author decided to use the **max pooling layer**.

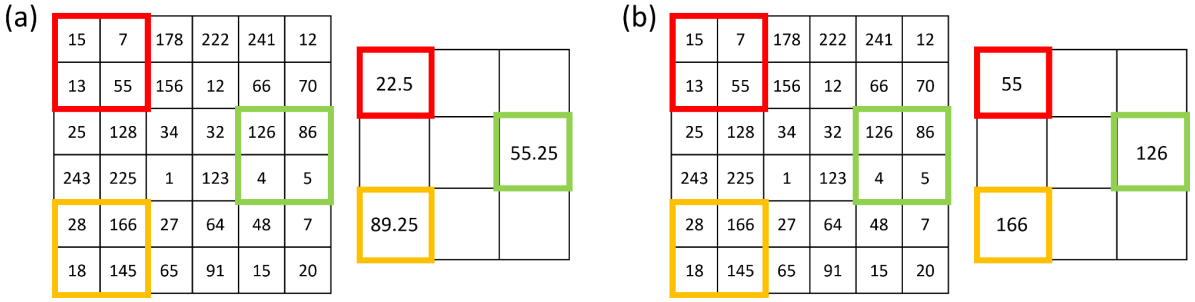

Figure 17. Pooling
(a) average pooling and (b) max pooling