A Mobile Application for Offline Handwritten Character Recognition

Thi Thi Zin, Moe Zet Pwint, Shin Thant
Graduate School of Engineering
University of Miyazaki
Miyazaki, Japan
thithi@cc.miyazaki-u.ac.jp, smithmoezet01@gmail.com, shinthant3010@gmail.com

Abstract— **The handwritten character recognition is a computerized system that is able to identify and recognize characters and words written by user. In this paper, we proposed offline handwritten character recognition using deep learning architecture. The Android application of the proposed system is also created by using OpenCV and TensorFlow Lite. The proposed system is aimed to use as a teaching aid in helping kindergarten to primary school level students, especially for practicing their writing and learning. The local handwritten dataset, which includes digit, English alphabet and mathematical symbols that are collected from students, is used for training and testing operations. According to the experimental results, the proposed system is very promising and it will be a useful application for educational environment.**

Keywords—handwritten character recognition, deep learning, android application, teaching aid

I. INTRODUCTION

The Convolutional Neural Network (CNN) is mostly applied in the handwritten recognition field due to its significant performance on the area of image classification, recognition and computer vision. Due to the emergence of E-learning in educational environment, the use of information technology (IT) as teaching aid is encouraging in children education [1]. One of the advantages of using E-learning is that the children can learn with their own study pace.

In this paper, we proposed an offline handwritten character recognition system that can be used as teaching aid application for kindergarten to primary level students. The remaining part of this paper is organized as follows. The section 2 describes proposed methods. The section 3 presented experimental results. Finally, the conclusions and future work are described in section 4.

II. PROPOSED METHODS

The proposed system is composed of three main steps: preprocessing, segmentation and classification. The overall system architecture is shown in figure 1. For the application development environment, we used Android studio (version 3.4.1) for creating Android application. The OpenCv for Android (opencv-3.4.6-android-sdk) is especially used for implementing image processing operations on Android. The TensorFlow (version 1.13.1) is used for developing the classification model. One of the advantages of using TensowFlow Lite framework is that training and testing operations can perform on high-end machine. The TensowFlow Lite converter generated '.tflite' (TensorFlow Lite model file) and this model file can be imported in Android to perform inference operation on the mobile devices.

A. Preprocessing

The characters or words written by user are stored as bitmap type for preprocessing step. Firstly the image are converted into grayscale image and then binary image. And then, the segmentation process is performed on the binary image for individual characters separation.

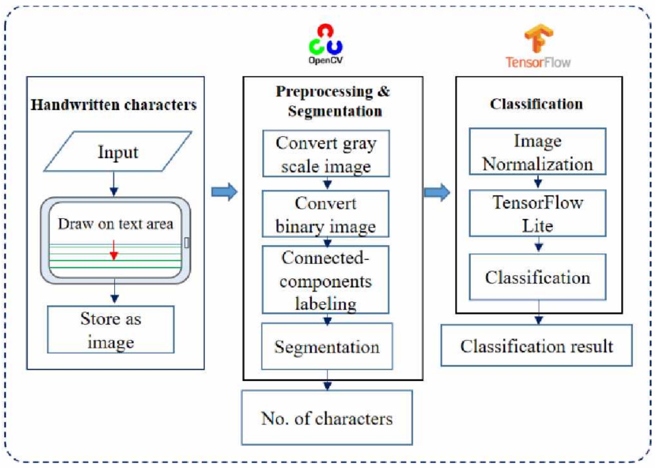


Fig. 1. Overall system architecture

B. Segmentation

In our proposed system, we perform segmentation using projection approach. We apply segmentation concepts of our previous research in [2]. Firstly, the input image is preprocessed by binarizing, complementing, thinning and cropping extra parts. Secondly, each connected component objects are labeled. Since some characters such as 'i' and 'j' contain more than one label, we combine the labels using three concepts for completely covered, partially covered and uncovered objects. Since there can be cursive words in labelled objects, thirdly we perform projection segmentation on each object. Then we finally apply two constraints which are closed character and pixel count detection to remove incorrect segmented projection points. The step by step segmentation process is described in figure 2. We use final segmented points to extract each characters from the input image.

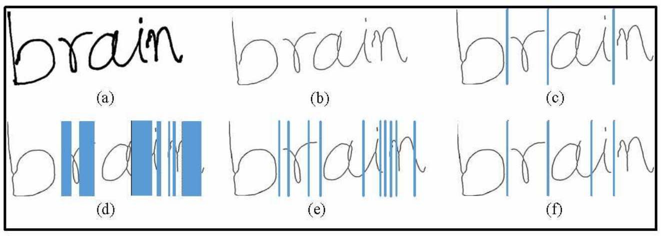


Fig. 2. (a) Original image (b) After preprocessing (c) Segmented lines after label separation (d) All possible projection segment points (e) Average projection lines (f) Segmented lines after over-segments removal

After segmentation, the sizes of individual character may vary in different image sizes. Therefore, image size normalization is done on each segmented characters with the size of $n \times n$ width and height. In our experiment, we set the normalized image size to 64×64 width and height.

C. Classification

After the segmentation process finished, the next step is to perform classification on each segmented character. The convolutional neural network (CNN) model is applied for character classification. The CNN architecture of the proposed system is shown in shown in table 1. To perform handwritten character classification on Android mobile phone and tablet, we applied TensorFlow Lite [3] model. The TensorFlow Lite model was set up as described in [4]. By using TensorFlow Lite converter, the generated file, '.tflite' file can able to perform inference process on phone and tablet.

TABLE I. PARAMETERS OF CNN'S ARCHITECTURE

Layer	Filter Size	Output Shape
Convolutional 1 + ReLU	$3\times3\times16$	$62 \times 62 \times 16$
Max Pooling 1	$1\times1\times16$	$31 \times 31 \times 16$
Convolutional 2 + ReLU	$3\times3\times32$	$29 \times 29 \times 32$
Max Pooling 2	$1\times1\times32$	$14 \times 14 \times 32$
Convolutional 3 + ReLU	$3\times3\times64$	$12 \times 12 \times 64$
Max Pooling 3	$1\times1\times64$	$6 \times 6 \times 64$

The three convolutional layer followed by Rectified Linear Unit (ReLU) are applied in the CNN. The filter size of 3×3 with stride value 1 is used in all layers with 16, 32 and 64 filters for each layer. After each convolution and ReLU, we performed down sampling process by max pooling operation with stride 2. The two fully connected layers are used and the first fully connected layer is composed of 80 nodes. The second fully connected layer or classification layer is composed with n nodes.

In our proposed system, different types of classifier are built for specific operations. The classifiers for digit and mathematical symbols, alphabets (including uppercase and lowercase) and combination of all types are created. Therefore, the classification layer has nodes of 18, 26, 26, 52 and 70 for digit and mathematical symbols, lowercase, uppercase, both lowercase and uppercase letters and combination of all types. The mathematical symbols include "+", "–", "×", "÷", "=", ">", "<" and "." (Decimal point). The classifiers with their respective ID are shown in table 2.

TABLE II. TYPES OF CLASSIFIERS

Classifier ID	Type of Classifier
1	Digit & Mathematical Symbols
2	Lowercase Letters
3	Uppercase Letters
4	Uppercase & Lowercase Letters
5	All types

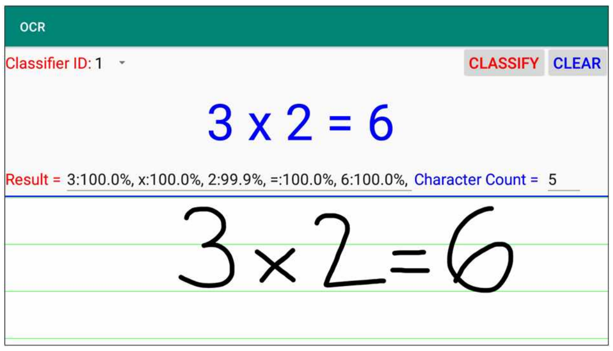
The reason behind creating different types of classifiers is to match the classifier type with educational content with higher classification rate. For example, the digit and mathematical symbols classifier will apply when students are practicing for writing digit and mathematical symbols.

III. EXPERIMENTAL RESULTS

The experiments are performed on our local dataset. The handwritten data are collected from primary level students. We performed training and testing operations on each types of classifiers and the number of training and testing images together with their respective accuracies are shown in table 3. After that, the generated classifiers are imported to perform inference operations on the mobile phone. Some experimental result using digit and mathematical symbols' classifier is shown in figure 3. Firstly, the user needed to select classifier ID for choosing type. The writing can be made on 4 green lines area and touching 'Classify' button for classification process. The classification result with accuracy and the number of characters are shown on the screen for information.

TABLE III. TRAINING AND TESTING ACCURACIES

Type of image	#Training	#Testing	Training acc. (%)	Testing acc. (%)
Digit and Maths Symbols	3425	900	99.6	96.6
Lowercase	4939	1300	98.8	94
Uppercase	4935	1300	99.1	97
Lowercase and uppercase	9874	2600	94.1	87.4
All type	13,299	3500	92.9	86.3



3 x 2 = 6

Result = 3:100.0%, x:100.0%, 2:99.9%, =:100.0%, 6:100.0%, Character Count = 5

3×2=6

Fig. 3. A illustration of digit and mathematical symbols classifier result

There are similarities between digit and alphabet such as '0' (digit) and 'o' (lowercase letter), '1' (digit) and 'l' (lowercase letter) and so on. When these similar characters are inputted to system by using 'All types' classifier (classifier ID 5), the classifier can sometime misclassify between these similar characters. Overall, the system experimental results are promising and it has good recognition rate on all classifiers.

IV. CONCLUSIONS

In this paper, we proposed an offline handwritten characters recognition system that can be used on Android mobile phones and tablets. The proposed system is aimed to apply as a teaching-aid for kindergarten to primary school level students to help in practicing their handwriting and it will be a useful application for educational environment. For the future work, we will make modification on the network model for having better classification result.

ACKNOWLEDGMENT

We would like to express our appreciation to Mr. Ogino Tsuhunobu, KJS Company LTD. for collaboration in this joint research.

REFERENCES

[1] Thi Thi Zin, Swe Zar Maw and Pyke Tin, "OCR Perspectives in Mobile Teaching and Learning for Early School Years in Basic Education", *In IEEE 1st Global Conference on Life Sciences and Technologies (LifeTech)*, pp. 173-174, Mar. 2019.

[2] Thi Thi Zin, Shin Thant and Ye Htet. "Handwritten Characters Segmentation using Projection Approach", *In IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech)*, pp. 107-108, Mar. 2020.

[3] https://www.tensorflow.org/lite (accessed on 1 July 2020).

[4] https://proandroiddev.com/mobile-intelligence-tensorflow-lite-classification-on-android-c081278d9961 (accessed on 1 July 2020).