Text/Non-Text Classification in Online Handwritten Documents with Recurrent Neural Networks

Truyen Van Phan and Masaki Nakagawa
Department of Electronic and Information Engineering
Tokyo University of Agriculture and Technology
Tokyo, Japan
{truyenvanphan}@gmail.com, {nakagawa}@cc.tuat.ac.jp

Abstract— In this paper, we propose a novel method for text/non-text classification in online handwritten document based on Recurrent Neural Network (RNN) and its improved version, Long Short-Term Memory (LSTM) network. The task of classifying strokes in a digital ink document into two classes (text and non-text) can be seen as a sequence labelling task. The bidirectional architecture is used in these networks to access to the complete global context of the sequence being classified. Moreover, a simple but effective model is adopted for the temporal local context of adjacent strokes. By integrating local context and global context, the classification accuracy is improved. In our experiments on the Japanese ink documents (Kondate database), the proposed method achieves a classification rate of 98.75%, which is significantly higher than the 96.61% in the previous work. Similarly, on the English ink documents (IAMonDo database), it produces a classification rate of 97.68%, which is also higher than other results reported in the literature.

Keywords- Text/Non-text classification; Text/Non-text separation; ink stroke classification; Recurrent Neural Networks; RNN; Long Short-Term Memory; LSTM

I. INTRODUCTION

In recent years, handwriting devices, pen-based or touch-based interfaces like smart phones, tablets, tablet PCs, electronic whiteboards, etc. have been used by many people. Moreover, electronic papers and paper-like PCs will be popular in near future. People are able to take notes, draw sketches and create diagrams. This kind of handwritten documents is natural to express ideas or concepts, and effective to brush them up. Due to the heterogeneous structures of text, drawings, diagrams, tables, etc., however, it brings new challenges to document analysis and recognition systems. Figure 1 shows examples of heterogeneous online handwritten documents from the Kondate database [1] and the IAMonDo database [2].

Although handwriting devices allow people to write text and non-text with a single pen or finger without changing any mode, the task of text/non-text classification is necessary for text recognition, text search and diagram interpretation. It is prerequisite to the selection of an appropriate engine to further process handwritten objects. From the classification results, text strokes can be sent to a handwriting recognizer or an ink search engine. On the other hand, non-text strokes can be grouped and recognized as higher level graphical entities like flow-chart, finite automata, etc. by a diagram interpreter.

Handwriting note taking is one of handwriting or pen-based applications that can exploit text/non-text classification and recognition techniques into reality. For real use, however, high performance is required. In other words, the results of text recognition or diagram interpretation should be not only highly accurate but also should be available in real-time, so that the processing time for text/non-text classification should be as little as possible. Since the publication of the Kondate and IAMonDo databases, text/non-text classification or mode detection has received increased interests from many researchers. Among them, the two methods proposed by Zhou et al. [3] and Delaye et al. [4] have established the state-of-the-art performances in Japanese and English documents, respectively. Zhou et al. [3] proposes an approach for classifying text and non-text in Japanese documents based on Markov random fields (MRFs), which integrates spatial interactions between neighboring strokes. Similarly, Delaye et al. [4] presents a system based on Conditional random fields (CRFs) that combines the different sources of context information (local, spatial, temporal) to improve the classification accuracy at the stroke level. In both methods, however, they use multiple support vector machine (SVM) classifiers to determine local, spatial and temporal potentials. Hence, the processing time becomes too long for practical use.

Recently, long short-term memory (LSTM) [5] has shown better performances compared to other algorithms for recognizing time-series patterns in several domains. It has been applied with remarkable success in the field of speech recognition [6] and handwriting recognition [7]. It is also used for text/non-text classification [8] and mode detection [9]. In [8], in order to employ the bidirectional LSTM neural network (BLSTM) to text/non-text classification, an input online handwriting pattern is not presented as a set of strokes, but as a stream of local feature vectors. The network recognizes and
translates it into a sequence of alphabet characters and non-text labels. Lastly, the sequence is then transcribed into a sequence of text and non-text labels. On the other hand, a standard recurrent neural network (RNN) and LSTM, an extension of RNN, are used for mode detection in [9]. The method outperforms other state-of-the-art classifiers with a classification rate of 98.47% and a classification speed of a few milliseconds. It is worth noting that the mode detection is different from the text/non-text classification. In the mode detection task, the input is an ink trace, and it is classified into text or non-text at the trace level. Meanwhile, in the text/non-text classification task, the input is an ink page of strokes, and it is classified at the stroke level. Hence, it is not comparable between the results of text/non-text classification task as ours and the results reported in [9].

In this paper, we propose to use the bidirectional RNN and LSTM networks for text/non-text classification. An input online handwriting pattern is presented as a sequence of feature vectors. However, it is different from the approach of [8] in two points: (1) the features are extracted globally on every single stroke, not locally on every point; (2) we employ the task of classification instead of the task of transcription so that every stroke is classified into text or non-text label. Moreover, since the bidirectional neural networks allow accessing to the global context in a document completely, we utilize the temporal local context to take advantage of the correlation between adjacent strokes so that the classification accuracy is improved.

The rest of this paper is structured as follows. Section II describes the classification method by using global context and local context models. Section III presents the feature extraction of single strokes and stroke pairs. Section IV introduces the two recurrent network architectures proposed in this work to be used for classification. Section V presents the experiments and their results. Finally, Section VI draws conclusions.

II. CLASSIFICATION METHOD

The text/non-text classification problem of ink strokes can be formulated as a sequence labelling problem. First, we start with a global context model to classify every stroke into two classes: text and non-text. Then, we integrate it with the local context of adjacent strokes to achieve better performance.

A. Global context model

The input is an ink document which consists of sequences of strokes. A stroke is a sequence of points recorded between a pen-down event and a pen-up event. A feature vector, denoted by $s$, is extracted on every stroke. The training samples consist of a set of N ordered strokes with feature vectors $s_n$, where $n = 1, ... N$ and class labels $l_n \in \{0, 1\}$ where $l_n = 1$ denotes a text stroke and $l_n = 0$ denotes a non-text stroke. For classification of single strokes, we train the network using the Back Propagation Through Time (BPTT). The output $y_n = y(s_n)$ of the resulting model represents the probability of a stroke being text given the feature vector $s_n$. The probability distribution of $l_n$ is a Bernoulli distribution given by:

$$P(l_n | s_n) = y_n^{l_n}(1 - y_n)^{1 - l_n} \quad (1)$$

Basically, the classification result in the global context model is determined by:

$$l_n = \begin{cases} 1 & \text{if } y_n \geq 0.5, \\ 0 & \text{otherwise}. \end{cases} \quad (2)$$

We call this classifier the single stroke classifier, binary classifier or global context model (GCM) classifier.

B. Local context model

In the global context model, the features of single strokes can give useful information for classification of text and non-text. Then, we expect to improve performance if we add the local context provided by adjacent strokes. For online ink, we have both temporal and spatial context. We focus on the use of temporal context of two adjacent strokes since it does not require any computation.

The basic idea of this model is to use the marginal distribution. Given two events $X$ and $Y$ whose joint distribution is known, the marginal distribution of $X$ is simply the probability distribution of $X$ averaging over information about $Y$. In other words, it is typically calculated by summing or integrating the joint probability distribution over $Y$.

$$P(X = x) = \sum_y P(X = x | Y = y)P(Y = y) \quad (3)$$

In our model, the marginal distribution of a stroke with feature vector $s_n$ is calculated by integrating the joint probability distribution of its preceding stroke or/and succeeding stroke. Three types of local context models are shown in Figure 2.

![Figure 2. Local context models based on the correlation between a stroke with its adjacent strokes.](image)

We call three models as preceding model (PM), succeeding model (SM) and bidirectional model (BM). The probability of a stroke being text correlated with its adjacent strokes in 3 models can be calculated by:

$$P_{PM}(l_n = 1 | s_n) = P(l_n = 1 | l_{n-1} = 1 | s_{n-1}, s_n)P(l_{n-1} = 1 | s_{n-1}) + P(l_n = 1 | l_{n-1} = 0 | s_{n-1}, s_n)P(l_{n-1} = 0 | s_{n-1}) \quad (4)$$

$$P_{SM}(l_n = 1 | s_n) = P(l_n = 1 | l_{n+1} = 1 | s_n, s_{n+1})P(l_{n+1} = 1 | s_{n+1}) + P(l_n = 1 | l_{n+1} = 0 | s_n, s_{n+1})P(l_{n+1} = 0 | s_{n+1}) \quad (5)$$

$$P_{BM}(l_n = 1 | s_n) = P_{PM}(l_n = 1 | s_n) + P_{SM}(l_n = 1 | s_n) \quad (6)$$

The probability of being non-text is computed similarly.

We now have the probabilities $P(l_n | s_n)$ which are obtained by using the single stroke classifier in the global context model. In order to obtain the probabilities $P(l_n | l_{n-1}, s_n, s_{n-1})$, we use a stroke pair classifier to determine the probabilities of a stroke pair belonging to three classes: text-text, text-nontext (equivalent with nontext-text) and nontext-nontext. Similarly with single stroke classification, we train this ternary classifier by using the network model.

The classification result in the local context model is then determined by:

$$l_n = \begin{cases} 1 & \text{if } P(l_n = 1 | s_n) \geq P(l_n = 0 | s_n), \\ 0 & \text{otherwise}. \end{cases} \quad (7)$$

We call this classifier the local context model (LCM) classifier.
C. Classifier combination

Since the combination of classifiers is often more accurate than a single classifier, we combine the classifier of the global context model with the classifier of the local context model. In this paper, we employ four basic combination rules as listed below:

The sum rule (SUM):
\[ l_n^* = \arg \max \{ \sum_{k=1}^{K} f_k(l_n|s_n), l_n \in \{0,1\} \} \] (8)

The product rule (PROD):
\[ l_n^* = \arg \max \{ \prod_{k=1}^{K} f_k(l_n|s_n), l_n \in \{0,1\} \} \] (9)

The max rule (MAX):
\[ l_n^* = \arg \max \{ \max_{k=1}^{K} f_k(l_n|s_n), l_n \in \{0,1\} \} \] (10)

The min rule (MIN):
\[ l_n^* = \arg \max \{ \min_{k=1}^{K} f_k(l_n|s_n), l_n \in \{0,1\} \} \] (11)

where \( K = 2, f_1(l_n|s_n) \) is the probability distribution of \( l_n \) calculated in the global context model (1) and \( f_2(l_n|s_n) \) is one of the three probability distributions of \( l_n \) calculated in the local context models (4), (5) and (6).

III. FEATURE EXTRACTION

For classifying an ink stroke into two classes, we use a set of contour-based shape features, spatial and temporal features since they have shown their effectiveness in text/non-text classification and mode detection. For classifying a pair of two adjacent strokes into three classes, we use a set of symmetrical measures between them. They are called unary features and binary features.

A. Unary feature

Two variants of feature sets are used to make two binary classifiers. A set of 11 features and another set of 19 features, which have been proposed in [3] and [4], respectively, are extracted from each stroke. The first variant contains contour-based shape features which can be divided into global and structural features. Global features use the dimensions of an ink stroke as principal features, such as length, area, compactness, etc. On the other hand, structural features describe relations between the path segments of an ink stroke, like curvature, perpendicular, etc. The second variant integrates six features of the spatial and temporal context information. These features are listed in Table 1.

B. Binary feature

We use four binary features to make a ternary classifier, which have been also already presented in [3], [4] and [10]. These features present the relationships between two neighboring strokes and they are extracted from every pair of two temporally adjacent strokes. The four binary features are: (1) the minimum distance between two strokes; (2) the minimum distance between the endpoints of two strokes, (3) the maximum distance between them and (4) the distance between the centers of the bounding boxes of two strokes. All of these features are symmetrical and independent from the temporal order of strokes.

C. Feature normalization

The extracted unary and binary feature values are causal variables so that we use the power transformation to make the density function closer to Gaussian. We set the power to 0.5 to transform each feature value. Furthermore, in order to standardize the feature values, we normalize the values of each feature based on the mean \( \mu_f \) and the standard deviation \( \sigma_f \) of that feature. The normalized feature value is then calculated by the following equation:

\[ v_f = \frac{v_f - \mu_f}{\sigma_f} \] (12)

The mean and standard deviation values are calculated for each feature on over the training samples. These values are stored and used for normalizing training, validation and testing samples.

TABLE I. TWO SETS OF FEATURES EXTRACTED FROM SINGLE STROKE AND LOCAL CONTEXT.

<table>
<thead>
<tr>
<th>Set1</th>
<th>Feature Description</th>
<th>Set2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stroke length</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Area of the convex hull of the stroke</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Compactness</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Eccentricity</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Ratio of the principal axis of the stroke seen as a cloud of points</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Rectangularity of the minimum area bounding rectangle of the stroke</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Circular variance of points of the stroke around the principal axis</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>Normalized centroid offset along the principal axis</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>Accumulated curvature</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>Accumulated squared perpendicularity</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>Accumulated signed perpendicularity</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>Stroke width</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>Stroke height</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>Number of temporal neighbors of the stroke</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>Number of spatial neighbors of the stroke</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Average of the distances from the stroke to spatial neighbors</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>Standard deviation of the distances from the stroke to spatial neighbors</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>Average of lengths of spatial neighbors</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>Standard deviation of lengths of spatial neighbors</td>
<td>19</td>
</tr>
</tbody>
</table>

IV. RECURRENT NEURAL NETWORKS

The task of classifying single strokes and stroke pairs in an ink document can be seen as a sequence labelling task. The task of labelling definitely depends not only on the preceding, but also on the succeeding input. Therefore, in this paper, the bidirectional architecture of the recurrent neural network and the recently developed long short-term memory are used to access to the future as well as the past context of the stroke sequence.

A. Recurrent neural network

Artificial neural networks (ANNs) are networks of small processing units or nodes, which are joined to each other by weighted connections. There are two basic types of ANNs: cyclic and acyclic. Acyclic ANNs do not allow cyclical connections, called feed-forward neural networks (FFNs). On the other hand, ANNs with cycles are referred to as feed-back, recursive or recurrent neural networks (RNNs). Figure 3 shows the architecture of a RNN.

In our architecture, the input layer size is the number of the unary or binary features. The number of hidden layers is...
two, and they consist of $k$ and $l$ units using the hyperbolic tangent (tanh) function for activation. The output layer size is two (text and non-text) in case of the single stroke classifier (SSC) and three (text-text, non-text-non-text and text-nontext) in case of the stroke pair classifier (SPC). The output activation functions used for RNN are the same as the standard artificial neural networks: logistic sigmoid for binary-classification and soft-max for multi-classification.

![Figure 3. Architecture of a RNN.](image)

**B. Bidirectional recurrent neural network**

The basic idea of a bidirectional recurrent neural network (BRNN) [11] is to present each training sequence forwards and backwards to two separate recurrent hidden layers, both of which are connected to the same output layers, as shown in Figure 4. Consequently, the network is able to incorporate contextual information both from past input and future input while the standard RNN is only able to incorporate it from past input.

![Figure 4. Architecture of a BRNN.](image)

**C. Long short-term memory network**

LSTM network [5] is an improved extension of the RNN. It overcomes the main problem of the traditional RNN, vanishing gradient problem. Instead of simple nodes, the hidden layers are made up of so-called LSTM blocks. In each block, three gates are used to store and access the data collected from the rest of the network. The gates are activated with the logistic sigmoid function $(1 + e^{-x})^{-1}$ (with range $[0,1]$).

**V. Evaluation**

**A. Data preparation**

To evaluate the performance of the proposed method and compare it with previous works, experiments using the same benchmark for the task of text/non-text classification have been performed on two heterogeneous online documents. For Japanese online documents, we have experimented on the Kondate database [1] which contains 669 heterogeneous pages acquired from 67 writers, 10 pages by each writer except 9 pages by the last writer. In [3], 310 pages were used for training classifiers and 359 pages were used for testing. In our experiments, since we need a dataset for validation, we use 210 pages for training, 100 pages for validation and the same 359 pages for testing. For English online documents, we have experimented on the IAMonDo database [2] which consists of 1000 heterogeneous pages produced by 200 writers. From the database, 403 pages are used for training, 200 pages for validation and 203 for testing. Table 2 summarizes the number of documents and strokes for training, validation and testing of the system, as well as the proportion of text (T) and non-text (N) strokes.

<table>
<thead>
<tr>
<th>Database</th>
<th>Subset</th>
<th># pages</th>
<th># strokes</th>
<th>% T</th>
<th>% N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kondate</td>
<td>Training</td>
<td>210</td>
<td>41,190</td>
<td>83.53</td>
<td>16.47</td>
</tr>
<tr>
<td></td>
<td>Validation</td>
<td>100</td>
<td>18,525</td>
<td>84.89</td>
<td>15.11</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td>359</td>
<td>71,846</td>
<td>85.44</td>
<td>14.56</td>
</tr>
<tr>
<td>IAMonDo</td>
<td>Training</td>
<td>403</td>
<td>143,350</td>
<td>80.96</td>
<td>19.04</td>
</tr>
<tr>
<td></td>
<td>Validation</td>
<td>200</td>
<td>68,726</td>
<td>83.60</td>
<td>16.40</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td>203</td>
<td>70,927</td>
<td>81.26</td>
<td>18.74</td>
</tr>
</tbody>
</table>

**B. Network settings**

In the training of the binary and ternary classifiers, we use four-layered networks with two fully connected recurrent hidden layers. The number of units/blocks on these two layers is set to 10 and 30 without attempt to optimize it. The learning rate on the training of BRNN and BLSTM are $10^{-4}$ and $10^{-4}$, respectively. The momentum rate is 0.9. For testing, each experiment has been repeated 20 times to keep results stable and independent from the random weight initialization. Our experiments on BRNN and BLSTM were performed with the open source software library RNNLIB [12].

**C. Experimental results**

On both datasets, for each type of network architectures, two single stroke classifiers (11-feature classifier and 19-feature classifier) and one stroke pair classifier are trained. We refer to these classifiers as SSC11, SSC19 and SPC. We trained three BRNN classifiers and three BLSTM classifiers for each dataset, denoted by SSC11_RNN, SSC19_RNN, SPC_RNN, SSC11_LSTM, SSC19_LSTM, and SPC_LSTM.

1) **Evaluation of single stroke classifiers**

In the global context model, the strokes can be classified by using a binary classifier which is trained by BRNN or BLSTM. Table 3 shows the text/non-text classification results of these single stroke classifiers. The computational time is the total time for classification without that for feature extraction on the entire testing dataset, 359 pages in Kondate and 203 pages in IAMonDo.

From the results, we can see that the BLSTM classifiers behave more stable than the BRNN classifiers. The differences between the maximum and minimum accuracy of the BLSTM classifiers are smaller than those of the BRNN classifiers about 1 point. For instance, in the SSC19 classifiers for Kondate, the difference of the BLSTM classifier is 1.94 points (95.68 ~ 97.62%) while that of BRNN is 2.8 points (94.65 ~ 97.45%). The BLSTM classifiers achieve better accuracy although they are about 4 times slower than the
BRNN classifiers. The BLSTM classifier with employing 19 features achieves good accuracy on both databases. For the Kondate database, the mean accuracy of 97.01% and even 97.62% (max) is significantly better than the result of 96.61% by the MRFs in [3]. As for the IAMonDo database, it achieves 96.93% (mean) and 97.34% (max). The mean accuracy is better than the accuracy of 96.66% by the CRFs in [10] and close to the accuracy of 97.01 by the BLSTM using local features in [8]. Moreover, the max accuracy of these classifiers is even better than the accuracy of 97.23% by the state-of-the-art method in [4].

2) Evaluation of stroke pair classifiers

In order to classify strokes in the local context model, stroke pair classifiers are trained by using BRNN or BLSTM to classify pairs of strokes into the three classes: text-text (TT), text-nontext (TN) and non-text-nontext (NN). The performances of these classifiers are shown in Table 4.

3) Evaluation of LCM classifiers

To evaluate the classifiers in the local context model, we select the single stroke classifiers and the stroke pair classifiers which produce the maximum accuracies. In each type of training network, the single stroke classifier SSC is integrated with the stroke pair classifier SPC to form a new classifier. There are three local context models for the stroke pair classifier: preceding, succeeding and bidirectional. Hence, we have three LCM classifiers: preceding-model classifier (PMC), succeeding-model classifier (SMC) and bidirectional-model classifier (BMC) for each pair of the feature set (11 or 19) and the training network (BRNN or BLSTM). Table 5 reports the classification results obtained by using these classifiers.

From the results, it is clear that the accuracy of the classifier which uses both the preceding and succeeding information in the bidirectional model is better than the one which uses only the preceding or the succeeding information. Except for the 19-feature classifiers for IAMonDo, the accuracies of the bidirectional-model classifiers are better than the GCM classifier. Especially, on Kondate, accuracies of 98.18% and 98.43% are achieved, being significantly improved when compared with the accuracies of the GCM classifiers.

4) Evaluation of combined classifiers

In this experiment, we measure the effect of combining the GCM classifier SSC and the LCM classifier BMC of which the 11 or 19 features are extracted and the classifiers are trained by using BLSTM. Table 6 shows the results of the experiment.

As the same with the single stroke classifiers, the LSTM classifiers achieve better accuracy but about 4 times slower than the RNN classifiers. It should be noted that on the IAMonDo database, the detection of text-nontext pairs is low although the detection of text-text pairs is very high. The reason of this result is that all the features are related to the reason of this result is that all the features are related to the...
5) Evaluation of computational complexity

We implemented the experiments on Intel® Core™ i7-4770 CPU 3.40GHz. On the entire testing dataset using the MIN(SSC19, BMC|LSTM) classifier, the extraction of 19 features costs 4.6 seconds, the classification by the GCM and the LCM classifiers costs 3 seconds and the combination costs just a few milliseconds. The total time for classifying ink strokes in 203 pages of the IAMonDo database is 7.6 seconds. Hence, the time required for processing an average document page from the test dataset is about 38 milliseconds. It is significantly faster than the time of 1.53 seconds reported by Delaye et al. [4]. The result is similar on the Kondate database.

6) Qualitative result

Figure 5 presents examples of text/non-text classification results on the Kondate and IAMonDo databases obtained by using the 19-feature GCM classifier SSC19_LSTM and the best classifier MIN(SSC19, BMC|LSTM), which is the combined classifier of the above SSC19_LSTM with the bidirectional LCM classifier by the min-rule as mentioned above. It shows that the combination of the local context with the global context has reduced errors.

![Figure 5](image)

Figure 5. Examples of text/non-text classification results (text strokes are black, non-text strokes are green, and misclassified strokes are red). The page a(1) and a(2) is from Kondate, while b(1) and b(2) is from IAMonDo. a(1) and b(1) are classified results by SSC19_LSTM while a(2) and b(2) are by MIN(SSC19, BMC|LSTM).

VI. CONCLUSION

This paper proposes a novel method for text/non-text classification in online handwritten documents based on RNN and LSTM. Our method takes the advantage of the global context from the bidirectional architecture of these networks. Moreover, the classified result is fixed by making use of the local context from the relationship between stroke and its adjacent strokes. From the integration of local context and global context, the classification accuracy is improved significantly. The proposed method achieves a classification rate of 98.75% on the Kondate database of the Japanese digital ink documents and 97.68% on the IAMonDo database of the English digital ink documents, which outperform other state-of-the-art methods in the literature.

The proposed method achieves an extremely high accuracy (nearly 99%) on the Kondate database, but it still remains challenges on the IAMonDo database, especially on the extracted diagram subset of this database. In this subset, the number of non-text strokes is larger than the number of text strokes. We plan to investigate other global features to improve the accuracies of the single stroke classifiers and stroke pair classifiers. More weighted combination rules will be tried to enhance the overall accuracy. Furthermore, we also attempt to utilize the local feature and employ the strategy of mode detection to make the task of text/non-text classification run in real-time.

ACKNOWLEDGMENT

This research is being supported by Grant-in-Aid for Scientific Research under the contract number (B) 24300095 and (S) 25220401. The authors would like to thank Mr. Cuong Tuan Nguyen and Mr. Anh Duc Le for giving helpful comments and suggestions.

REFERENCES


