Recognition of Anomalously Deformed Kana Sequences in Japanese Historical Documents

Nam Tuan Ly(a), Kha Cong Nguyen(b), Nonmembers, Cuong Tuan Nguyen(c), Member, and Masaki Nakagawa(d), Fellow

SUMMARY This paper presents recognition of anomalously deformed Kana sequences in Japanese historical documents, for which a contest was held by IEICE PRMU 2017. The contest was divided into three levels in accordance with the number of characters to be recognized: level 1: single characters, level 2: sequences of three vertically written Kana characters, and level 3: unrestricted sets of characters composed of three or more characters possibly in multiple lines. This paper focuses on the methods for levels 2 and 3 that won the contest. We basically follow the segmentation-free approach and employ the hierarchy of a Convolutional Neural Network (CNN) for feature extraction, Bidirectional Long Short-Term Memory (BLSTM) for frame prediction, and Connectionist Temporal Classification (CTC) for text recognition, which is named a Deep Convolutional Recurrent Network (DCRN). We compare the pretrained CNN approach and the end-to-end approach with more detailed variations for level 2. Then, we propose a method of vertical text line segmentation and multiple line concatenation before applying DCRN for level 3. We also examine a two-dimensional BLSTM (2DBLSTM) based method for level 3. We present the evaluation of the best methods by cross validation. We achieved an accuracy of 89.10% for the three-Kana-character sequence recognition and an accuracy of 87.70% for the unrestricted Kana recognition without employing linguistic context. These results prove the performances of the proposed models on the level 2 and 3 tasks.

key words: historical documents, deformed kana recognition, handwriting recognition, deep neural networks

1. Introduction

Until the Edo period (1603–1868), Japanese documents were vertically written with a brush or wood-block printed. Characters, especially Kana (phonetic characters made from Kanji of Chinese origin), were deformed anomalously and cursively written as shown in the following sections, so even experts have difficulty reading them. Due to the demand for preserving historical documents and availing them for research without damaging physical documents, digitization and preservation of digital reproductions have been studied and practiced in many regions and languages [1]–[6].

The Center for Open Data in the Humanities (CODH) in Japan is studying and developing ways to enhance access to Japanese humanities data and constructing data platforms to promote collaborative research among people with diverse backgrounds. Under the support by CODH, Pattern Recognition and Media Understanding (PRMU) held a contest to read anomalous Kana in 2017 [7]. The tasks are divided into three levels in accordance with the number of characters in a circumscribed rectangle: level 1: single characters, level 2: sequences of three vertically written Kana characters, and level 3: unrestricted sets of characters composed of three or more characters possibly in multiple lines.

Here we summarize some publications reporting historical document processing in the languages of Chinese origin written with brushes. Kim et al. developed a system for digitizing more than 10 million Hanja* historical documents [8]. To build the system, they employed manual typing and handwriting recognition based on the Mahalanobis distance. In China, Digital Heritage Publishing Ltd. digitized more than 36,000 volumes (4.7 million pages) of Siku Quanshu, which is the largest collection of books on Chinese history compiled by 361 scholars during the Qianlong period (1711–1799). They first applied optical character recognition (OCR) to segment and recognize characters and then manually corrected misrecognized characters [9]. Kitadai et al. reported a system to help archeologists read wooden tablets excavated from ancient ruins [10]. Given an input character image, the system provides functions to restore the image and presents similar character images already decoded, using simple pattern matching since the purpose is to nominate candidates and sample patterns are very limited. Truyen et al. [11] developed a system for digitizing hundreds of thousands of Nom historical documents. Nom is the old Vietnamese writing system composed of original Chinese characters and Vietnamese characters created in the same way as Chinese characters, i.e., formed from radicals. The digitization system segments a document image into characters and recognizes individual characters by the modified quadratic discriminant function (MQDF) [12]. To train MQDF, pattern augmentation was applied.

Next, we will briefly survey handwritten document recognition. Most traditional methods are based on pre-segmentation of a text line image into characters, especially for Japanese and Chinese [13], [14]. However, this pre-segmentation is costly and error prone because segmentation directly affects the performance of the whole system.

---

*Hanja is the Korean name for Chinese characters incorporated into the Korean language with Korean pronunciation.
On the other hand, segmentation-free methods can avoid the segmentation errors. They have been employed only for western handwritten documents on the basis of the Hidden Markov Model (HMM) [15], [16] so far, but first Recurrent Neural Networks (RNNs) and then a Connectionist Temporal Classification (CTC) have outperformed HMM in sequence prediction and labeling tasks [17], [18]. More recently, Deep Neural Networks (DNNs) combined with CTC have been proposed and proven to be very effective for offline handwriting text recognition. A. Grave et al. combined Multi-Dimensional LSTM (MDLSTM) and CTC to build an end-to-end trainable model for offline handwritten Arabic recognition [19]. B. Shi et al. proposed an end-to-end trainable model combining CNN for feature extraction and LSTM to recognize printed scene text [20]. S. Rawls et al. also used an end-to-end trainable CNN + LSTM model for English and Arabic handwritten text recognition [21]. Their models work for languages with a small character set. Applications of DNNs were limited until recently for languages with a large character set. R. Messina et al. presented MDLSTM and CTC to recognize offline Chinese handwritten text using raw features, rather than those extracted from CNN [22]. More recently (in 2017) Y. Chao et al. presented CNN and MDLSTM with CTC for handwritten Chinese text recognition [23]. At the same time, we also presented a combination of CNN and LSTM with CTC for recognizing offline handwritten Japanese text [24] and historical Japanese text [25]. Then, we proposed an end-to-end method for ordinary Japanese text [26]. H. Yang et al. employed CNN or CNN + LSTM, both followed by CTC for Chinese text recognition in historical documents [27]. D. Valy et al. used CNN and 2D LSTM to recognize Khmer historical palm leaf manuscripts [28]. They all recognize text in single line images. This paper is based on a previous conference paper [25], but omits level 1 and focuses on levels 2 and 3 in more detail. Moreover, we added end-to-end-training and a 2DBLSTM after the contest. We compare the pre-trained Convolutional Neural Network (CNN) approach and the end-to-end approach with more detailed variations for level 2: recognizing sequences of three vertically written Kana characters. Then, we propose a method of vertical text line segmentation and multiple line concatenation before applying the DCRN model for level 3: recognizing unrestricted sets of characters in multiple lines. We also examine two-dimensional Bidirectional Long Short-Term Memory (2DBLSTM)-based methods for level 3 and compare their performances with the vertical text line segmentation based method.

The rest of this paper is organized as follows: Section 2 describes the overview of the contest. Section 3 presents methods for recognizing sequences of three vertically written Kana characters (level 2). Section 4 describes methods for recognizing unrestricted sets of Kana characters (level 3). Section 5 concludes the paper and mentions future work.
Fig. 3  Similar notations between different categories.

Fig. 4  Fade and show-through.

Fig. 5  Fragmented patterns and noisy patterns.

Fig. 6  Various backgrounds.

similiar to a notation of the category ‘ka’ as shown in Fig. 3. The different notations and similar notations between different categories are difficult problems for recognizing the old handwritten Kana. Since the original images are scanned from old Japanese books, they are fade and show-through as shown in Fig. 4; smeared and stained as shown in Fig. 5. Their backgrounds are often neither uniform nor even as shown in Fig. 6.

3. Three Kana Sequence Recognition

This section presents recognition methods and evaluations on the level 2 dataset.

3.1 Level 2 Dataset

The level 2 dataset consists of 79,165 images of vertically written text lines composed of three Kana. Since PRMU did not publicize the test set of the contest, we reserve one of the 15 pre-modern books as the test set to evaluate the models. Among the 15 books, the 15th book contains many fragmented and noisy patterns as well as various backgrounds. Thus, the 15th book is selected as the test set for levels 2 and 3. That is, all level 2 images and all level 3 images in this book are used for testing in the experiments for levels 2 and 3, respectively. The other books are divided randomly to form the training and validation sets with the ratio of 9:1. Consequently, the level 2 dataset consists of three subsets: the training set consisting of 56,097 images, the validation set consisting of 6,233 images, and the testing set consisting of 16,835 images. Figure 7 shows some vertical text line images in the dataset.

3.2 Methods for Level 2

Many segmentation-free methods based on DNNs [19], [20] have been proposed for image recognition tasks. In level 2, we employ a hierarchical architecture of a feature extractor by a CNN, a frame predictor by BLSTMs, and a decoder by a CTC to recognize three-Kana-character sequence images as shown in Fig. 8. We named this architecture a Deep Convolutional Recurrent Network (DCRN).

In the DCRN, the CNN feature extractor, which is usually pretrained by single-character images as in level 1, extracts the sequence of features for all the frames from an input text line image, where each frame is a region within the input image from which features are extracted by CNN. Then, BLSTM frame predictor predicts a list for character labels with scores (label distribution) for each frame. Finally, the CTC decoder finds the most probable label sequence using the forward and backward algorithms.

Long Short-Term Memory (LSTM) is an RNN architecture designed to receive an input sequence with long-range dependencies and output another sequence that has one-to-one correspondence to the input sequence [29]. The hidden units of RNN are replaced by ‘memory cell’ units, which can store and retrieve information over time, giving them access to long-range context. Each memory cell has three multiplicative gate units: the input gate, the forget gate and the output gate to control, respectively, the write, erase, and read access operations to the unit. These control gates can be shared among cells. A group of cells sharing common control gates form a block of LSTM cells.

Bi-directional LSTM allows access to the context of an input from both forward (left to right) and backward (right
Fig. 8 Network architecture of DCRN.

to left) directions. It consists of two LSTM layers that scan the input in both the directions [19].

CTC is an algorithm designed for sequence labeling tasks where it is difficult to segment an input sequence to segments that exactly matches those in a target sequence. CTC performs alignment of a probability output sequence to a given label sequence. As a result, the system does not need to segment an input sequence for training. The probability of a label sequence $l$ from an input sequence $x$ is the total probability of all the paths $\pi$ that produce the label sequence as shown in Eq. (1):

$$p(l|x) = \sum_{\pi \in \pi_l} p(\pi|x)$$ (1)

CTC loss is the total negative log likelihood $-\ln p(l|x)$ over all pairs of an input sequence $x$ and a target label $l$ from training patterns.

3.3 Implementation Details

For the contest, we proposed the pretrained CNN approach of the DCRN model and outperformed the other 9 participants to win the contest. After the contest, however, we proposed the end-to-end approach of the DCRN model. The difference between the pretrained CNN approach and the end-to-end approach is that the former pretrains the CNN by isolated character patterns before it is used to extract a feature sequence from a text line image. On other hand, the end-to-end approach does not pretrain the CNN network but trains the weights of CNN and those of BLSTM on pairs of images and sequences by only one loss function. Sections 4.3.1 and 4.3.2 describe the implementation details of these approaches.

In both approaches, we binarize all images using Otsu’s method [30] and scale them into the same 64-pixel width while maintaining the aspect ratio. The Otsu’s method can remove background noise due to smears, stains, fade and show-through and so on, but some noise remains. The CNN feature extractor can extract key features while ignoring this remaining noise.

3.3.1 Pretrained CNN Approach

We employ a cascade of five blocks of a convolutional layer and a max-pooling layer followed by two full-connected layers to make the CNN component for feature extraction. The detailed architecture of our CNN model is given in Table 1 in which ‘maps,’ ‘k,’ ‘s,’ and ‘p’ denote the number of kernels, kernel size, stride, and padding size of convolutional layers, respectively.

Firstly, the CNN model is pretrained by the training set in the level 1 dataset using the stochastic gradient descent with the learning rate of 0.001 and the momentum of 0.95 (Hereafter, training or pretraining is made by the training set in some dataset). We apply mini-batch training with the batch size of 64 samples. After training the CNN model, we remove just the softmax layer or both the full connected layers and the softmax layer from the CNN model to use the remaining network as the feature extractor. Although the CNN architecture is the same, there are three methods to extract features from an input text line image by the CNN model.

The first method slides a sub-window of $64 \times 64$ pixels through the text line image with a sliding stride size (overlap sliding) of 12 or 16 pixels and applies the CNN network without the softmax layer to extract features. We call this method DCRN-o_12 and DCRN-o_16 when the sliding stride size is 12 and 16 pixels, respectively. Figure 9 shows this way of forming the feature extractor.

The second method employs a sub-window of $64 \times 32$ pixels and the sliding stride size of 32 pixels (without overlap sliding) as shown in Fig. 10 and applies the CNN network without the softmax and full connection layers to ex-

<table>
<thead>
<tr>
<th>Table 1 Network configuration of our CNN model.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>Input</td>
</tr>
<tr>
<td>Convolution - ReLu</td>
</tr>
<tr>
<td>MaxPooling</td>
</tr>
<tr>
<td>Convolution - ReLu</td>
</tr>
<tr>
<td>MaxPooling</td>
</tr>
<tr>
<td>Convolution - ReLu</td>
</tr>
<tr>
<td>MaxPooling</td>
</tr>
<tr>
<td>Convolution - ReLu</td>
</tr>
<tr>
<td>MaxPooling</td>
</tr>
<tr>
<td>Full-connected - ReLu</td>
</tr>
<tr>
<td>Softmax</td>
</tr>
</tbody>
</table>
extract features from the input image. The full connection layer is further removed because an input image has a different size from character images used to pretrain the CNN model. We call this method DCRN-wo.

The third method does not use the sliding window but directly uses the text line image as an input of the CNN model and applying the CNN network again without the softmax and full connection layers to extract features for the same reason as in DCRN-wo. Figure 11 shows its architecture. We call this method DCRN-ws.

For the frame predictor, we employ three levels of BLSTM networks with each level composed of two LSTMs (forward and backward), where every LSTM contains 128 memory blocks with each block having a single cell. A fully connected layer and a softmax layer with the node size equal to the character set size \( n = 47 \) are applied after each time step of the frame predictor. Here, the number of categories is increased by one to include the blank character. The classifier is trained using the online steepest descent with the learning rate of 0.0001 and the momentum of 0.9. All vertical text line images in the dataset are scaled to the same width before being fed to the system.

3.3.2 End-to-End Approach

The end-to-end approach does not pretrain the CNN network but trains the weights of the CNN and those of BLSTM on pairs of images and sequences by only one loss function. We employ the CNN network without the fully connected and softmax layers for the same reason as in DCRN-wo and DCRN-ws. To reduce the training time of this approach, we apply the batch normalization [31] after each convolutional layer in the CNN network.

Table 2 shows the architecture of the CNN network used in the convolutional feature extractor, where ‘maps,’ ‘k,’ ‘s,’ and ‘p’ denote the number of kernels, kernel size, stride and padding size of each convolutional layer, respectively. The architecture consists of four convolutional layers. Batch normalization and Max-Pooling are applied after each convolutional layer. The Leaky ReLu [32] activation function is employed in all convolutional layers.

At the frame predictor, we employ the same Deep BLSTM network as in the pretrained CNN approach. To prevent overfitting when training the model, the dropout (dropout rate = 0.2) is also applied in each layer in Deep BLSTM. The fully connected layer and the softmax layer the same as the pretrained CNN approach are applied after each time step of Deep BLSTM. The end-to-end DCRN model is trained using the stochastic gradient descent with the learning rate of 0.001 and the momentum of 0.9. The training process stops when the recognition accuracy on the validation set does not gain after 10 epochs.

3.4 Experiments for Level 2

The performance on levels 2 and 3 is measured in terms of Label Error Rate (LER) and Sequence Error Rate (SER), which are defined as follows:

\[
\text{LER} = \frac{\text{Mistakes}}{\text{Total Characters}}
\]

\[
\text{SER} = \frac{\text{Mistakes}}{\text{Total Lines}}
\]
Table 3. Recognition error rates (%) on level 2 dataset.

<table>
<thead>
<tr>
<th>Networks</th>
<th>LER Valid set</th>
<th>LER Test set</th>
<th>SER Valid set</th>
<th>SER Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCRN-wo</td>
<td>14.19</td>
<td>26.79</td>
<td>33.51</td>
<td>59.28</td>
</tr>
<tr>
<td>DCRN-ws</td>
<td>10.21</td>
<td>18.56</td>
<td>25.07</td>
<td>44.81</td>
</tr>
<tr>
<td>DCRN-o_16</td>
<td>9.72</td>
<td>14.44</td>
<td>23.62</td>
<td>35.11</td>
</tr>
<tr>
<td>DCRN-o_12</td>
<td>8.65</td>
<td>12.88</td>
<td>21.03</td>
<td>31.60</td>
</tr>
<tr>
<td>End-to-End DCRN-ws</td>
<td>3.10</td>
<td>10.90</td>
<td>13.10</td>
<td>27.70</td>
</tr>
</tbody>
</table>

\[\text{LER}(h, S') = \frac{1}{2} \sum_{(x,z) \in S'} \text{ED}(h(x), z) \quad (2)\]

\[\text{SER}(h, S') = \frac{100}{|S'|} \sum_{(x,z) \in S'} \begin{cases} 0 & \text{if } h(x) = z \\ 1 & \text{otherwise} \end{cases} \quad (3)\]

where \(S'\) is a testing set of input-target pairs \((x, z)\), \(h\) is a pattern classifier, \(Z\) is the total number of target labels in \(S'\), and \(\text{ED}(p, q)\) is the Levenshtein edit distance between two sequences \(p\) and \(q\).

Table 3 shows the performances for the five models. Comparison of DCRN-o_12 and DCRN-o_16 suggests that the smaller stride of the sliding window with overlap works better in the convolitional feature extractor. The result that DCRN-o_12 and DCRN-o_16 are better than DCRN-ws suggests that the convolutional feature extractor made by sliding a sub-window through an input image is superior to the convolutional feature extractor made by directly using an input text line image as the input of the CNN model. The DCRN-o_16 model was awarded the best method prize for achieving 87.6% recognition accuracy for Lv2, while other methods recorded an average of 45.6% recognition accuracy for the secret test set \([7]\). The worst network in Table 3 is DCRN-wo, which suggests sliding a sub-window without overlap may lose the information from the border regions in the sub-window when extracting features by CNN.

With a 10.90% LER and 27.70% SER, the End-to-End DCRN-ws obtained the best recognition accuracy. This result suggests that the end-to-end model approach works substantially better than the pretrained CNN approach.

On the other hand, there are still large gaps between the validation and testing sets. This suggests that the number of training samples was not adequate, so overfitting occurred. Employing more samples for training or applying data augmentation may decrease the error rates to some extent.

Figure 12 shows some correctly recognized and misrecognized samples by DCRN-o_12, whose sequence error rate is 31.60%. For each correctly recognized sample, the upper image is an input vertical text line composed of three Kana characters and the text below shows the recognition result (ground-truth). For each misrecognized sample, the upper image is an input image and the text below shows the ground-truth followed by “->” and the recognition result.

3.5 Cross Validation of End-to-End DCRN-ws

We employ the \(k\)-fold cross validation to evaluate the performance of the proposed End-to-End DCRN-ws model more fairly. Since the level 2 dataset is made from 2,222 scanned pages of 15 pre-modern Japanese books, we use the value of \(k = 5\) and split the dataset into 5 folds from 15 books with each fold having the same number of books. In other words, fold 1 consists of data from the 1st, 2nd, and 3rd books, fold 2 consists of data from the 4th, 5th, and 6th books, and so on. On the basis of the five folds, the \(i\)-th model of End-to-End DCRN-ws is trained and validated by four folds but not the \(i\)-th fold. These four folds are divided randomly to form the training and validation sets with the ratio of 9:1. Then, the validated \(i\)-th model is evaluated on the \(i\)-th fold. The average accuracy of the five models is calculated as follows:

\[\text{Avg}(\text{error rate}) = \frac{1}{5} \sum_{i=1}^{5} \text{error rate}_i \times N_i \quad (4)\]

where \(\text{error rate}_i\) is the error rate of the \(i\)-th model, \(N_i\) is the number of test images for the \(i\)-th model, and \(N\) is the total number of the test images for all models.

Table 4 shows the recognition error rates of the five models. On average, this approach achieved a 14.45% LER and 34.44% SER, but these results are inferior to those shown in Table 3. The reason seems to be that fewer patterns were used here for training than in the previous experiment.
whereas more patterns were used for testing. Another reason seems to be that the test patterns in the 15th book in the previous experiment were not the hardest to read. In fact, the worst error rate was recorded by Model 3, which employed the 7th, 8th, and 9th books for testing but others for training and validation.

Another observation can be made from Table 4. The results greatly vary because we prepared the folds on the basis of separate books. This way of preparing folds is fair and able to predict unseen books and characters. When the training set is not large, however, systems might be evaluated by very different patterns. This seems to be another reason for the inferior performance mentioned above. Increasing test patterns is the best method, but changing the preparation of folds should be considered, such as preparing the folds by sampling from all the books.

4. Unrestricted Kana Recognition

This section presents recognition methods and evaluations on the level 3 dataset.

4.1 Level 3 Dataset

The hardest task is level 3, which could be considered as an extension of level 2.

In level 3, 12,583 images are divided into three subsets: the training set of 10,118 images, the validation set of 1,125 images, and the testing set of 1,340 images as mentioned in Sect. 3.1. All images in level 3 consist of three or more Kana characters written on one vertical line or multiple vertical lines. In addition to the difficulties mentioned in level 2, there are some other difficulties such as the vertical and horizontal guide lines (Fig. 13), the overlap or even touching between two vertical lines (Fig. 14). In Fig. 14 (a), we draw blue bounding boxes to show each character.

4.2 Methods for Level 3

We propose three approaches for solving the level 3 task. The first approach applies vertical text line segmentation which segments multiple vertical text lines into individual vertical text lines and concatenates them to form a single vertical line image before employing the Kana sequence recognition of level 2. Since BLSTM for level 2 only works on a single vertical line image, we need to segment vertical text line images and reshape them into a single vertical text line image. The second approach employs the pretrained CNN network from level 2 and adds a 2DBLSTM that does not require any line segmentation to avoid the limitation of BLSTM in the first approach. The last approach employs only a 2DBLSTM. The second and the third approaches produce 2-dimensional predictions for a multi-line input image. The prediction is scanned and serialized into a prediction sequence and aligned with a label sequence for minimizing CTC loss. Thus, we can train the networks directly without needing any line segmentation.

Multi-dimensional LSTM is an extension of LSTM to n-dimensions by using n recurrent connections from the previous states along every dimension with n forget gates [19].

The idea of accessing bi-directional context by BLSTM can also be extended to multi-dimensional LSTM. For a two-dimensional LSTM, the bi-directional context of a 2D input along every dimension creates a total of four directions accessed by four layers of two-dimensional LSTM. We call a two-dimensional LSTM with bi-directional context access as 2DBLSTM. In our case, a 2DBLSTM receives a document image and outputs two-dimensional sequential predictions. In general, a multi-dimensional LSTM receives an n-dimensional input, scans it through each dimension as a sequential input and outputs another set of n-dimensional sequential predictions that have one-to-one correspondence to the input. CTC determines the final labels.

Common to the three approaches, in the same way as for level 2, we binarize all images using Otsu’s method [30] and scale them into the same width of 64 pixels while maintaining the aspect ratio.

Level 3, however, includes multiple-line images. In fact, 40.82% percent images have two lines. For such patterns, the above scaling implies each text line may only have half the width in such cases.

4.2.1 Vertical Text Line Segmentation and Concatenation Approach

The first approach segments vertical text lines and concatenates them into a single line before applying Kana sequence recognition.

(a) Overlap between two vertical lines. (b) Two samples of touching between two vertical lines.

Fig. 14 Overlap or touching between two vertical lines.
connected component has a bounding box. Some of them have widths larger than heights and vice versa, so we calculate the representative width of a component by the arithmetic mean of its width and height. We sort components in ascending order by their sizes and calculate the average size (AS) of all $N$ components in the page from the larger half of components since those in the smaller half are often noises and isolated strokes. Images that have widths less than AS are considered as one-line images and left for the subsequent step.

$$AS = \frac{2}{N} \sum_{i=1}^{N} S_i$$  \hspace{0.5cm} (5)$$

For images having widths equal or larger than AS, we employ our implementation [13] of the X-Y cut method [33] to separate them into text line images. The X-Y cut method calculates the vertical projection profile for each image and generates text-line borders at the transiting positions of non-zero projection to zero projection and zero projection to non-zero projection. The X-Y cut method sometimes overcuts text line images, so we combine the text line images that have widths less than half of AS.

Then, we apply the Voronoi diagram method [34] to segment images unsegmented by the X-Y cut method. A Voronoi diagram shows the borders between CCs. To adapt the method to our purpose, we calculate the direction of each Voronoi border from its start point and end point, where the start and the end points are the upper and the lower points of a Voronoi border, respectively. We discard borders extending to the left or the right side of images while keeping borders starting from the top and ending at the bottom of the images. If both the above methods are unsuccessful for segmenting text line images, which is judged by the width of a vertical text-line exceeding AS, we forcefully separate at centerlines of images. These cases often include text lines touching each other or horizontal guide lines.

Finally, we concatenate text line images from right to left and create a text line image from top to bottom by aligning the concatenated text line images with the center. Figure 15 shows some generated text line images.

(2) Kana sequence recognition
We employ the best model (End-to-End DCRN-ws) and second best model (DCRN-o.12) in level 2 for recognizing single-line images.

For training the two models, we apply the above vertical text line segmentation and concatenation to all training and validation images of level 3. Then, we train the two models using the training images until the recognition accuracy on the validation set does not gain after 10 epochs.

Moreover, we can also use the training images of level 2 for this approach. We denote the training images of level 3, which are the results of the vertical text line segmentation and concatenation, as STL_Lv3 (to denote single text line images of the level 3) and denote those of level 2, which are all single-line images, as STL_Lv2. Then, when End-to-End DCRN-ws and DCRN-o.12 are trained by STL_Lv3 alone, we call them Seg + End-to-End DCRN-ws_Lv3 and Seg + DCRN-o.12_Lv3, respectively. Moreover, when they are trained by both STL_Lv3 and STL_Lv2, we call them Seg + End-to-End DCRN-ws_Lv2&3 and Seg + DCRN-o.12_Lv2&3, respectively. We will compare their recognition performances in the evaluation.

4.2.2 CNN Plus 2DBLSTM Approach

The second approach employs a pretrained CNN network and a 2DBLSTM that does not require any vertical text line segmentation. We reuse the pretrained CNN network without the softmax and full connection layers from the pretrained CNN approach of level 2 described in Table 1 for feature extraction. The output of the pretrained CNN is scanned by two levels of the 2DBLSTM. The first level is composed of four LSTM layers that each have 64 single-cell memory blocks, and the second level is also composed of four LSTM layers, each having 128 single-cell memory blocks (2DBLSTM_b:64_b:128). We call this model CNN + 2DBLSTM_b:64_b:128. The output of the 2DBLSTM is scanned through the order of writing in the vertical direction (top to bottom, right to left) and is then aligned to the sequence of character labels for training using the CTC layer.

For training the networks, we use level 2 and level 3 images. All images of levels 2 and 3 are scaled into the same width of 64 pixels including two-line images. This means that single text line images are scaled with their width being 64 pixels whereas text line images of two lines are scaled so that each text line has the width of almost 32 pixels. To help the model learn these scaled characters, we add the level 2 images scaled to the width of 32 pixels to the training set. We denote this model as CNN + 2DBLSTM_b:64_b:128_Lv3 when it is trained by the level 3 dataset only and as CNN + 2DBLSTM_b:64_b:128_Lv2&3 when it is trained by the level 2 and level 3 datasets.

4.2.3 2DBLSTM Approach

The third approach replaces the CNN in the second approach by a stage of 2DBLSTM with the result of three
Table 5  Recognition error rates (%) on level 3 dataset.

<table>
<thead>
<tr>
<th>Networks</th>
<th>Valid set</th>
<th>Test set</th>
<th>Valid set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN + 2DBLSTM b:64 b:128 Lv3</td>
<td>14.45</td>
<td>44.18</td>
<td>55.82</td>
<td>97.16</td>
</tr>
<tr>
<td>CNN + 2DBLSTM b:64 b:128 Lv2&amp;3</td>
<td>23.59</td>
<td>43.09</td>
<td>67.38</td>
<td>94.55</td>
</tr>
<tr>
<td>2DBLSTM b:2 b:10 b:50 Lv3</td>
<td>18.59</td>
<td>46.73</td>
<td>66.37</td>
<td>98.81</td>
</tr>
<tr>
<td>2DBLSTM b:2 b:10 b:50 Lv2&amp;3</td>
<td>15.55</td>
<td>37.72</td>
<td>63.35</td>
<td>94.55</td>
</tr>
<tr>
<td>Seg + DCRN-o_12 Lv3</td>
<td>11.72</td>
<td>26.70</td>
<td>49.14</td>
<td>82.57</td>
</tr>
<tr>
<td>Seg + DCRN-o_12 Lv2&amp;3</td>
<td>9.47</td>
<td>24.24</td>
<td>46.53</td>
<td>78.81</td>
</tr>
<tr>
<td>Seg + End-to-End DCRN ws Lv3</td>
<td>4.30</td>
<td>18.50</td>
<td>21.50</td>
<td>73.70</td>
</tr>
<tr>
<td>Seg + End-to-End DCRN ws Lv2&amp;3</td>
<td>2.80</td>
<td>12.30</td>
<td>15.40</td>
<td>54.90</td>
</tr>
</tbody>
</table>

stages overall. The output by the three stages of 2DBLSTM is then scanned and aligned to the sequence of character labels using CTC in the same way as in the second approach.

To reduce the number of time steps in each dimension of 2DBLSTM, inputs are scanned through a window of consecutive time steps in each dimension. Weighted connections are used to transform each window of consecutive time steps to a single input time step.

We use the three stages of 2DBLSTM. Each stage is composed of four LSTM layers. There are 2, 10 and 50 single-cell memory blocks for each layer in the first, second, and third stages, respectively. The input window sizes are $2 \times 2$, $4 \times 2$, and $4 \times 2$ for this sequence of stages, respectively. The input window sizes are applied to reduce the dimensionality of input images. Two feedforward fully-connected layers with 6 hidden units for the first layer and 20 hidden units for the second layer are applied between the first and second stages and between the second and third stages, respectively. The feedforward layers merge the output of previous stages before their subsequent stages. They reduce the number of weights compared with directly connecting the stages.

For training the networks, we use the level 2 and level 3 images in the same way as in the previous approach. We denote this model as 2DBLSTM b:2 b:10 b:50 Lv3 when it is trained by the level 3 dataset only and as 2DBLSTM b:2 b:10 b:50 Lv2&3 when it is trained by the level 2 and level 3 datasets.

### 4.3 Experiments on Level 3

Table 5 shows the recognition error rates of the three approaches for level 3. When only the level 3 dataset is used for training, Segmentation + End-to-End DCRN ws Lv3 achieves the best results: 18.50% LER and 73.70% SER. When both level 2 and level 3 datasets are used for training, Segmentation + End-to-End DCRN ws Lv2&3 again achieves the best results: 12.30% LER and 54.90% SER. In both cases, the vertical text line segmentation and concatenation approach outperforms the CNN plus 2DBLSTM approach as well as the 2DBLSTM approach without CNN.

Table 5 also shows that training with both the level 2 and level 3 datasets improves the recognition accuracy for all approaches. For the CNN plus 2DBLSTM approach and the 2DBLSTM approach, we halved the width of the level 2 training patterns in order to add them to train the models, although this width reduction may have had side effects. The results show its effect probably because the large set of 56,097 characters contributes to learning these models.

Figure 16 shows some samples correctly recognized and misrecognized by Seg + DCRN-o_12 Lv3, which had a 26.70% character error rate. The Seg + DCRN-o_16 Lv3 model was awarded the best method prize for achieving 39.1% recognition accuracy for Lv3, while other methods recorded an average of 21.5% recognition accuracy for the secret test set [7]. For each correctly recognized sample, the upper image is an input and the text below shows the recognition result (ground-truth). For each misrecognized sample, the upper image is an input image and the text below shows the ground-truth followed by “->” and the recognition result.

![Correctly recognized samples](image1)

![Misrecognized samples](image2)
Table 6  Recognition error rates (%) of five models.

<table>
<thead>
<tr>
<th>Models</th>
<th>LER</th>
<th>SER</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>24.19</td>
<td>75.73</td>
<td>3,433</td>
</tr>
<tr>
<td>Model 2</td>
<td>16.66</td>
<td>66.44</td>
<td>1,353</td>
</tr>
<tr>
<td>Model 3</td>
<td>55.84</td>
<td>87.25</td>
<td>2,709</td>
</tr>
<tr>
<td>Model 4</td>
<td>19.76</td>
<td>51.14</td>
<td>617</td>
</tr>
<tr>
<td>Model 5</td>
<td>18.72</td>
<td>75.57</td>
<td>4,471</td>
</tr>
<tr>
<td>Average</td>
<td>23.73</td>
<td>75.95</td>
<td>-</td>
</tr>
</tbody>
</table>

4.4 Cross Validation of Seg Plus End-to-End DCRN_ws

In the same way as in Sect. 3.5, we employ the five-fold cross validation to fairly evaluate the performance of the proposed methods on the level 3 dataset. We prepare five folds and select training/validation sets for each test hold the same way as above.

Table 6 shows the recognition error rates of the five models. On average, this approach achieved a 23.73% LER and 75.95% SER, but these results are inferior to those of the model trained by only the level 3 dataset shown in Table 5. This result is the same as that in Sect. 3.5. Large variation in the performance is again the same as that in Sect. 3.5, possibly for the same reason.

5. Conclusion

This paper compared several Deep Neural Network architectures to recognize anomalously deformed Kana Sequence in Japanese historical documents in accordance with two levels (2 and 3) in a contest held by IEICE PRMU 2017. For level 2, the end-to-end approach achieved the best Label Error Rate (LER) of 10.90% and Sequence Error Rate (SER) of 27.70%. For level 3, the vertical text line segmentation and concatenation approach achieved the best LER of 13.30% and SER of 54.90% when trained by both the level 2 and level 3 datasets. The sequence error rate is so high that linguistic context must be incorporated. For cross validation experiments, organization of folds for cross validation should be reconsidered for better prediction of error rates.

For future work, we will apply the language statistics and context processing to improve the accuracy of the systems. Moreover, we hope to extend the research to cover Kanji characters. To improve the performance, we need to apply data augmentation.

Acknowledgments

We thank Prof. Kitamoto in Japan and the committee members of the IEICE PRMU contest for preparing the datasets and leading this contest. This work is partially supported by the Grant-in-Aid for Scientific Research (S)-18H05221 and (A)-18H03597 as well as ROIS-DO-JOINT no. 027RP2018.

References


Kha Cong Nguyen was born on 8 December 1989 in Viet Nam. He received his B.Eng. degree in Comp. Eng. from Hanoi Univ. of Science and Technology in 2012 and M.Sc. degree in Comp. Sci. from TUAT in 2017. Since April 2017, he has been a Ph.D. student in the Dept. of Electronic and Information Eng. at TUAT. He is currently working on offline handwritten Japanese text recognition and historical document processing.

Cuong Tuan Nguyen was born on 21 March 1988 in Vietnam. He received his B.Eng. degree in Comp. Eng. from Hanoi Univ. of Science and Technology in 2012 and M.Sc. and Ph.D. in Comp. Sci. from TUAT in 2014 and 2017. Since April 2017, he has been working as an assistant professor at TUAT. He is currently working on handwritten text recognition and writer identification.

Masaki Nakagawa was born on 31 Oct. 1954. He received his B.Sc. and M.Sc. degrees from the Univ. of Tokyo in 1977 and 1979. During the academic year 1977/78, he enrolled in the computer science course at Essex Univ. in the UK and also received a M.Sc. with distinction in Computer Studies from there in July 1979. He received his Ph.D. from Univ. of Tokyo in Dec. 1988. He has been working at TUAT since April 1979. He is currently a professor of media interaction in the Department of Computer and Information Sciences.

Nam Tuan Ly was born on 23 October 1989 in Vietnam. He received his B.Eng. degree in Comp. Eng. from Hanoi Univ. of Science and Technology in 2013 and M.Sc. degree in Comp. Sci. from TUAT in 2018. Since Oct. 2018, he has been a Ph.D. student in the Dept. of Electronic and Information Eng. at TUAT. He is currently working on offline handwritten text recognition of Japanese and historical document recognition.

Kha Cong Nguyen was born on 8 December 1989 in Viet Nam. He received his B.Eng. degree in Comp. Eng. from Hanoi Univ. of Science and Technology in 2012 and M.Sc. degree in Comp. Sci. from TUAT in 2017. Since April 2017, he has been a Ph.D. student in the Dept. of Electronic and Information Eng. at TUAT. He is currently working on offline handwritten Japanese text recognition and historical document processing.

Cuong Tuan Nguyen was born on 21 March 1988 in Vietnam. He received his B.Eng. degree in Comp. Eng. from Hanoi Univ. of Science and Technology in 2012 and M.Sc. and Ph.D. in Comp. Sci. from TUAT in 2014 and 2017. Since April 2017, he has been working as an assistant professor at TUAT. He is currently working on handwritten text recognition and writer identification.

Masaki Nakagawa was born on 31 Oct. 1954. He received his B.Sc. and M.Sc. degrees from the Univ. of Tokyo in 1977 and 1979. During the academic year 1977/78, he enrolled in the computer science course at Essex Univ. in the UK and also received a M.Sc. with distinction in Computer Studies from there in July 1979. He received his Ph.D. from Univ. of Tokyo in Dec. 1988. He has been working at TUAT since April 1979. He is currently a professor of media interaction in the Department of Computer and Information Sciences.