

# Attempts to recognize anomalously deformed Kana in Japanese historical documents

Hung Tuan Nguyen, Nam Tuan Ly, Kha Cong Nguyen, Cuong Tuan Nguyen, Masaki Nakagawa  
Department of Computer and Information Sciences  
Tokyo University of Agriculture and Technology  
2-24-16 Naka-cho, Koganei-shi, Tokyo, 184-8588 Japan  
{ntuanhung, namlytuan, congkhanguyen, ntcuong2103}@gmail.com, nakagawa@cc.tuat.ac.jp

## ABSTRACT

This paper presents methods for three different tasks of recognizing anomalously deformed Kana in Japanese historical documents, which were contested by IEICE PRMU<sup>1</sup> 2017. The tasks have three levels: single character recognition, three Kana characters sequence recognition and unrestricted Kana recognition. We compare several methods for each task. For the level 1, we evaluate CNN based methods and BLSTM based methods. For the level 2, we consider several variations of a combined architecture of CNN and BLSTM. For the level 3, we compare an extension of the method for the level 2 and a segmentation based method. We achieve the single character recognition accuracy of 96.8%, the three Kana characters sequence recognition accuracy of 87.12% and the unrestricted Kana recognition accuracy of 73.3%. These results prove the performance of CNN and BLSTM on these tasks.

## CCS CONCEPTS

• Applied computing → Optical character recognition

## KEYWORDS

historical document, Kana recognition, handwriting recognition, Deep neural networks.

## ACM Reference format:

H. T. Nguyen, N. T. Ly, K. C. Nguyen, C. T. Nguyen and M. Nakagawa. 2017. Attempts to recognize anomalously deformed Kana in Japanese historical documents. In *Proceedings of ICDAR Workshop, Kyoto, Japan, November 2017 (HIP'17)*, 6 pages. <https://doi.org/10.1145/3151509.3151514>

<sup>1</sup> Technical Committee on Pattern Recognition and Media Understanding (PRMU) of the Inst. of Electronics, Information and Communication Engineers (IEICE), Japan

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).  
HIP2017, November 10–11, 2017, Kyoto, Japan  
© 2017 Association for Computing Machinery.  
ACM ISBN 978-1-4503-5390-8/17/11\$15.00  
<https://doi.org/10.1145/3151509.3151514>

## 1 INTRODUCTION

Japanese use Kanji, ideographic characters of Chinese origin, and Kana, phonetic characters made from Kanji characters. Kanji are used for nouns and stems of verbs, adverbs, adjectives etc. while Kana are used for conjugation parts and so on. There are thousands of Kanji characters but only 46 Kana characters since Japanese has 5 vowels and 10 consonants, thus 50 phonemes, but 4 phonemes were merged to others.

Until the Edo period (1603-1868), Japanese documents were vertically written with brush or wood-block printed. Characters, especially Kana, were deformed anomalously and cursively written as shown in the following sections so that even experts have difficulties to read them. Due to the demand of preserving historical documents and availing them for research without damaging physical documents, digitization and preservation of digital reproduction has been studied and practiced in many regions and languages [1-9]. Center for Open Data in the Humanities (CODH) in Japan is studying and developing for enhancing access to Japanese humanities data and constructing data platforms to promote collaborative research among people with diverse backgrounds. Under the support by CODH, PRMU has held a contest to read anomalous Kana in 2017 [10].

This paper presents several methods for each task. It is organized as follows: Section 2 describes the tasks of the contest in more detail. Section 3 presents methods based on CNN and BLSTM for the task 1 of single Kana recognition. Section 4 presents methods for the task 2 of recognizing sequences of vertically written three Kana characters. Section 5 describes methods for the task 3 of unrestricted sets of Kana characters. Section 6 draws our conclusion.

## 2 RECOGNITION TASKS IN THE CONTEST

The competition is divided in three different levels i.e., level 1: recognition of segmented single characters, level 2: recognition of sequences of vertically written three Kana characters and level 3: recognition of unrestricted sets of characters composed of three or more than three characters possibly in multiple lines [11]. Figure 1 shows a sample page and examples of the three levels. All the tasks require to recognize Kana characters rather than Kanji. The tasks are provided by CODH. Their open

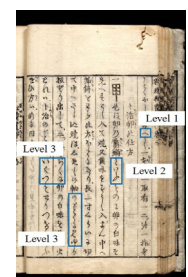


Figure 1: Sample page.

dataset stores 403,242 character images of 3,999 categories [12]. Each character is annotated with its bounding box and Unicode.

### 3 SINGLE KANA RECOGNITION

#### 3.1 Level 1 dataset

The dataset for the level 1 consists of 228,334 single Kana images segmented from 2,222 scanned pages of 15 historical books provided by CODH. Among these books, the 15<sup>th</sup> book contains many fragmented and noisy patterns as well as various backgrounds. Thus, the 15<sup>th</sup> book is selected for forming the test set for the level 1, level 2 and level 3. Namely, all single Kana images, vertical Kana text line images and unrestricted Kana characters images in this book are used for testing. Single Kana images from other books are divided randomly to make the training set and the validation set with the ratio of 9:1. Consequently, the level 1 dataset consists of three subsets: the training set of 162,323 images, the validation set of 18,061 images and the testing set of 47,950 images.

Each image of the level 1 belongs to one of 46 Kana categories. As in other handwriting databases, there are large deformations and variations in the patterns of the same category. Moreover, the old Kana uses several different notations for the same category, such as shown in Figure 2, where the category ‘o’ has two notations and the category ‘ni’ has four notations. Furthermore, a notation of category ‘u’ is similar to a notation of category ‘ka’ as shown in Figure 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 the old Japanese books, the backgrounds are sometimes heavily noised and badly damaged as shown in Figure 4 and they are not uniform as shown in Figure 5.

Another difficulty comes from the imbalanced categories where some categories have more occurrences while some other categories have less occurrences. For instance, the category ‘ni’ has 15,982 occurrences while the category ‘e’ has only 440 occurrences in the dataset. The occurrences of characters are different depending on the sources.

#### 3.2 Methodology for level 1

We consider the level 1 as an image classification task. The deep neural networks have currently become a state-of-the-art approach for the image classification task, especially the Convolutional Neural Network (CNN) and the Long-Short Term Memory (LSTM) are the main architectures. We employ both of the two architectures CNN and LSTM for solving this task.

For preprocessing images, we employ the Otsu binarization method [13] for reducing the complexity of the background and enhancing the foreground as the first step in preprocessing. Then, we apply the padding by white pixels to place the original image in the center and the linear spatial normalization as the second step in order to rescale all images into a unique size due to the requirement of CNN. Then, the padded image is linearly rescaled into a 64-by-64 pixel image. The third step is standardization all images so that the pixel values have zero-mean and unit variance. Figure 6 shows the results of the preprocessing steps.

After the preprocessing steps, we also employ the various transformations including rotation, shearing and shifting with random parameters on training patterns whenever they are fed into the training system, in order to generalize the training data and avoid the imbalanced categories.

Next, we present the details of CNN and LSTM used for this task. We employ and modify four different CNNs which are the established networks for general image classification.

The first network (AlexNet-based\_c:5) is derived from AlexNet [14] which consists of 5 convolutional layers and 3 max pooling layers after the first, second and fifth convolutional layers. Three fully connected layers are used for processing the extracted features from the above convolutional layers. The second network (VGGNet-based\_c:13) is deeper than AlexNet-based\_c:5 with 13 convolutional layers which are divided into four blocks as presented in [15]. Besides that, we also employ VGGNet-based\_c:16 with 16 convolutional layers divided into four blocks. The fourth network (ResNet-based\_c:150) consists of 150 layers which are divided into 50 blocks with 3 convolutional layers for each block [16]. Moreover, inside each block, there is a residual connection between the first and the last layer of that

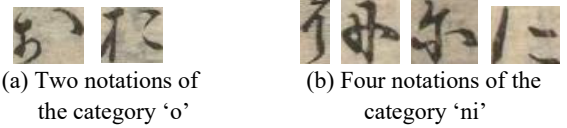


Figure 2: Different notations of the same category.

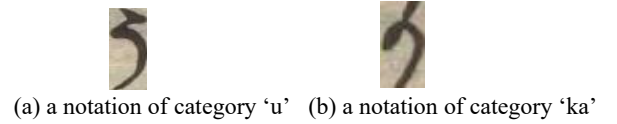


Figure 3: Similar notations between different categories.



Figure 4: Fragmented patterns and noisy patterns.



Figure 5: Various backgrounds.

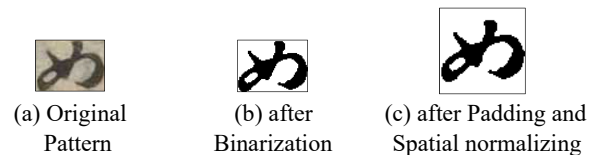


Figure 6: Results of preprocessing steps.

block. The fifth network (HighwayNet-based\_c:16) with 16 gated convolutional layers in which a gate is used to control the data flow coming in and going out each layer [17]. The control gates of HighwayNet-based\_c:16 are useful for training the deeper networks since they can preserve the backward gradients from the last layer back to the first layer so that not only the later layers but the first layers are trained. For each of the above five networks, we employ three fully connected layers whose size are 4096, 4096 and 46 respectively. We apply dropout for the two first fully connected layers for improving the performance of CNNs.

Beside the five CNNs, we also employ three networks of 2-Dimensional Bidirectional LSTM (2DBLSTM) in which each 2DBLSTM layer consists of 4 unidirectional LSTM layers with different directions along two dimensions of images [18]. The single layered 2DBLSTM network consists of one 2DBLSTM layer of 16 units (2DBLSTM\_u:16) with 29,166 parameters. The two layered 2DBLSTM is constructed by the first 2DBLSTM layer of 4 units and the second of 20 units (2DBLSTM\_u:4\_u:20) with 29,918 parameters. The three layered 2DBLSTM is composed of three 2DBLSTM layers of 4, 20, 100 units, respectively (2DBLSTM\_u:4\_u:20\_u:100) with 557,038 parameters. Since the input image for the three 2DBLSTM networks could have different size, we do not employ the padding and spatial normalization step of the preprocessing while preparing data for training these 2DBLSTM networks. The recurrent connections of 2DBLSTM layers represent the dependencies between the pixels of an image which are important for obtaining a good representation.

### 3.3 Experiments for level 1

During the training process of all the above networks, we apply the stochastic gradient descent algorithm with the momentum of 0.9, the learning rate of 0.001 and the early stop scheme after 20 epochs without improving accuracy on the validation set. After the training process, we select the best parameters of the network by the best accuracy on the validation set. For speeding up the training process of CNNs, we use GPUs and apply mini-batch training with size of 50. Finally, we employ a voting-based ensemble method which uses the recognition results from many trained networks. Table 1 shows recognition rates for the level 1 by different networks. The voting-based ensemble of CNNs achieves the recognition accuracy of 96.8%, which is much better than the single CNNs.

The binarization step is important in preprocessing since it is useful to increase the recognition accuracy. The improvements for the CNNs are only about 1 point (percentage difference), which implies that CNNs have already learned the good features even if original images contain noises and non-uniform backgrounds. On the other hand, the 2DBLSTM networks achieve the increase of more than 20 point by binarization. It suggests that the 2DBLSTM networks require the processing steps to remove noises and make the uniform background. Among them, the three layered 2DBLSTM network is not better than the other 2DBLSTM networks when trained without binarization, since it has a larger number of parameters than the others.

**Table 1: Recognition rates (%) on test set of level 1.**

Networks	Accuracy (%)	Preproc. images w/o binarization	Preproc. images	Preproc. images & transf.
AlexNet-based_c:5		93.42	94.66	94.79
VGGNet-based_c:13		94.75	95.67	95.74
VGGNet-based_c:16		94.60	95.92	95.96
ResNet-based_c:150		91.01	91.75	95.15
HighwayNet-based_c:16		94.98	95.18	95.29
2DBLSTM_u:16		51.48	73.06	-
2DBLSTM_u:4_u:20		55.02	87.73	-
2DBLSTM_u:4_u:20_u:100		43.18	89.81	-
Ensemble of CNNs		96.01	<b>96.80</b>	97.13
Ensemble of 2DBLSTM		55.10	90.21	-

*Preproc. and transf. denote preprocessed and transformations.*

The performance of ResNet-based\_c:150 is not as good as the other CNNs since the number of patterns in dataset is not large enough to train its 150 convolutional layers.

## 4 THREE KANA SEQUENCE RECOGNITION

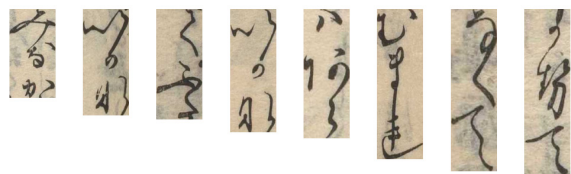
### 4.1 Level 2 dataset

The level 2 dataset consists of 92,813 images of vertical text lines composed of three Kana. It is split into three subsets: the training set consisting of 56,098 images, the validation set consisting of 6,233 images and the testing set consisting of 13,648 images. The details for dividing three subsets has been presented in Section 3.1. Figure 7 shows some vertical text line images in the dataset.

### 4.2 Methodology for level 2

In recent years, many segmentation-free methods based on Deep Neural Networks [19, 20] have been studied and have demonstrated to be powerful in image-based sequence recognition tasks. In the level 2, we employ a combined architecture of CNN, BLSTM, and CTC to recognizing three Kana character sequence images. We named this architecture as Deep Convolutional Recurrent Network (DCRN) [21]. The architecture of DCRN consists of 3 components of a convolutional feature extractor, recurrent layers, and a transcription layer, from bottom to top as shown in Figure 8.

From the bottom of the DCRN, the convolutional feature extractor, which is usually pretrained by single characters as in the level 1, extracts the sequence of features from an input text line image. Then, the recurrent layers predict each frame of the feature sequence output by the convolutional feature extractor. At the top of the DCRN, the transcription layer translates the pre-frame predictions by the recurrent layers into the final label sequence.



**Figure 7: Some vertical text line images in the dataset.**

### 4.3 Implementation details

We employ a cascade of 5 blocks of a convolutional layer followed by a max-pooling layer and two full-connected layers to make the CNN component used in the convolutional feature extractor. The detailed architecture of our CNN model is listed in Table 2 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 dataset of the level 1 using the stochastic gradient descent with the learning rate of 0.001 and the momentum of 0.95. We apply mini-batch training with the batch size of 64 samples. After training the CNN model, we remove just the softmax layer or remove both the full connected layers and the softmax layer from the CNN model to use the remaining network as the convolutional feature extractor. Although the architecture of CNN is the same, there are three ways to extract features from an input text line image by the CNN model.

The first way slides a sub-window of 64x64 pixels through the text line image with 12 pixels or 16 pixels of the sliding stride size (overlap sliding) and applies the remaining CNN network without the softmax layer to extract features. We call the former DCRN-o\_12 and the latter DCRN-o\_16, respectively.

The second way employs a sub-window of 64x32 pixels and the sliding stride size of 32 pixels (without overlap sliding). We call this DCRN-wo.

The third way does not use the sliding window, but directly use the text line image as an input of the CNN model. Figure 9 shows its architecture. We call this DCRN-ws. DCRN-wo and DCRN-ws apply the CNN without both the full connected layers and the softmax layer removed as the convolutional feature extractor because the size of an input image is different from the size of character images used to pretrain the CNN models.

At the recurrent layers, we employ 1-Dimensional Bidirectional LSTM (1DBLSTM) network with 128 nodes of three layers. The recurrent layers and the transcription layer are trained using the online steepest decent with the learning rate of 0.0001 and momentum of 0.9. All of the vertical text line images in the dataset are scaled to the same width before used to train the system.

### 4.4 Experiments for level 2

In order to evaluate the performance on the level 2 and level 3, the performance is measured in terms of Label Error Rate (LER) and Sequence Error Rate (SER) that are defined as follows:

$$\text{LER}(h, S') = \frac{1}{Z} \sum_{(x, z) \in S'} \text{ED}(h(x), z)$$

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

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 performance by 4 models. Comparison of DCRN-o\_12 and DCRN-o\_16 implies that the smaller stride of

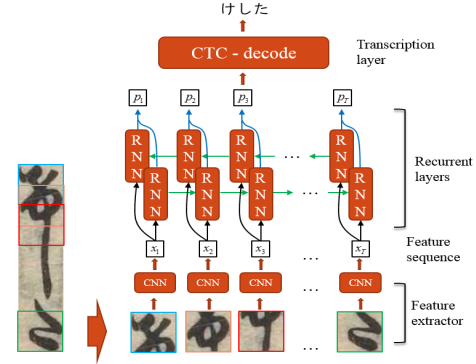


Figure 8: Network architecture of DCNR.

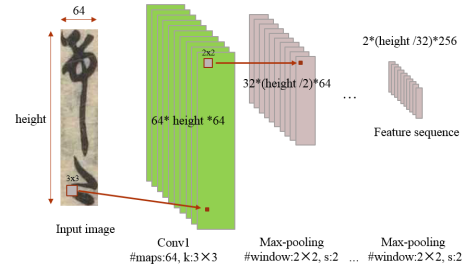


Figure 9: Convolutional feature extractor in DCRN-ws.

Table 2: Network configuration of our CNN model.

Type	Configurations
Input	64×64 image
Convolution - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling	#window:2×2, s:2
Convolution - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling	#window:2×2, s:2
Convolution - ReLu	#maps:128, k:3×3, s:1, p:1
MaxPooling	#window:2×2, s:2
Convolution - ReLu	#maps:128, k:3×3, s:1, p:1
MaxPooling	#window:2×2, s:2
Convolution - ReLu	#maps:256, k:3×3, s:1, p:1
MaxPooling	#window:2×2, s:2
Full-connected - ReLu	#nodes:200
Full-connected - ReLu	#nodes:200
Softmax	#nodes: 46(number class)

Table 3: Recognition error rates (%) on test set of level 2.

Networks	LER		SER	
	Valid set	Test set	Valid set	Test set
DCRN-o_12	8.65	<b>12.88</b>	21.03	<b>31.60</b>
DCRN-o_16	9.72	14.44	23.62	35.11
DCRN-wo	14.19	26.79	33.51	59.28
DCRN-ws	10.21	18.56	25.07	44.81

the sliding window with overlap works better in the convolutional 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 input image is superior to the convolutional feature extractor made by directly using the input text line image as the input of the CNN model. The worst network in Table 3 is DCRN-wo, which implies sliding sub-window without overlap may lose the information of the border region in the sub-window when extracting features by CNN. There are a total of 4,312 misrecognized samples among



13,648 samples. Most of them are misrecognizing only one character among three characters. Figure 10 shows some misrecognized samples by DCRN-o\_12 whose sequence error rate is 31.60%.

## 5 UNRESTRICTED KANA RECOGNITION

### 5.1 Level 3 dataset

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

In the level 3, there are 12,583 images which are divided into three subsets, the training set of 10,118 images, the validation set of 1,125 and the testing set of 1,340 images as mentioned in Section 3.1. All images in the level 3 consists of more than three Kana characters which are written on one vertical line or multiple vertical lines (almost two vertical lines). In addition to the difficulties mentioned in the level 1 and the level 2, there are some other difficulties such as the vertical and horizontal guide lines (Figure 11), the overlap or even touching between two vertical lines (Figure 12) and fade and show-through (Figure 13). In Figure 12(a), we draw the blue bounding boxes to help to see each character.

### 5.2 Methodology for level 3

We propose three approaches for solving the level 3. The first approach applies vertical line segmentation and multiple line concatenation before employing Kana recognition of the level 2. Since 1DBLSTM only works on one line, we need to segment vertical lines and reshape the multiple vertical line images into a single vertical line image. The second approach employs 2DBLSTM which does not require any pre-segmentation to avoid the limitation of 1DBLSTM in the first approach. The last approach uses an object detection CNN [22] with pre-trained CNNs from the level 1 which does not require any pre-segmentation.

#### 5.2.1 Vertical line segmentation method

Figure 14 shows the process to segment text lines and concatenate them to a single text line. From an image of multiple vertical text lines, we locate connected components (CCs). Since there are lots of noises after binarization in historical documents, we remove CCs with their areas smaller than the threshold of 25 pixels (5x5). We consider them as noisy components. The size of the  $i$ -th connected component ( $S_i$ ) is calculated from the average of height and width of its bounding box. We sort components in the ascending order by their sizes and calculate the average size ( $AS$ ) of the whole  $N$  components in the page from the larger half of components since those in the smaller half are often noises and isolated strokes.

$$AS = \frac{2}{N} \sum_{i=\frac{N}{2}}^N S_i$$

Images with their widths less than  $AS$  are considered as one line images and left for the next step. For images having widths equal or larger than  $AS$ , we employ the X-Y cut method [23] to separate them into text lines. The X-Y cut method calculates the vertical projection profile for each image and generate text-line borders at

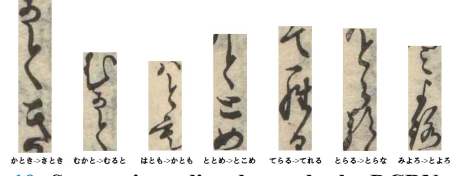


Figure 10: Some mispredicted samples by DCRN-o\_12.

the transiting positions of non-zero projection to zero projection and via versa. The X-Y cut method sometimes makes overcutting of text lines, so that we combine the text lines whose widths are less than the half of  $AS$ .

Then, we apply the Voronoi diagram method to segment images unsegmented by the X-Y cut method [23]. Voronoi diagram shows the borders between CCs. To fit the method to our purpose, we calculate the direction of each Voronoi border from its starting point and endpoint. We discard borders extending to the left side or the right side of images while keep borders starting from top and ending at the bottom of the images. If both of the above methods are unsuccessful for segmenting text lines whose widths are equal or larger than  $AS$ , we forcefully separate at centerlines of images. These cases often include text lines touching each other or horizontal guided lines.

Finally, we concatenate text lines from right to left and create a sequential text-line from top to bottom. Text lines in the sequential text-line are aligned to the center of the text-line. This generated text line is used for the method present in Section 4. Figure 15 shows some generated sequential text-lines.

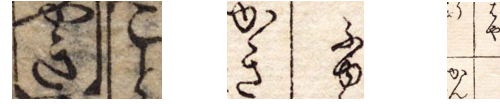


Figure 11: Level 3 images containing the vertical and horizontal guide lines.

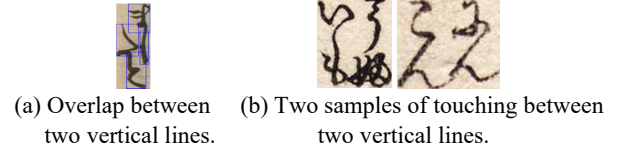


Figure 12: Overlap or touching between two vertical lines.

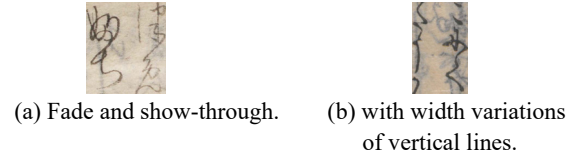


Figure 13: Fade and show-through.

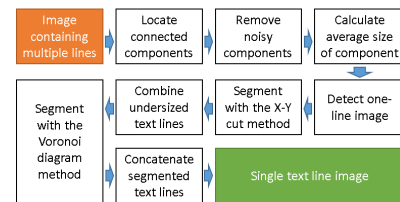


Figure 14: A sequential line made from the multiple lines.



Figure 15: Concatenated text lines.

After segmenting and concatenating text lines to single lines, we use the best model of DCRN-o\_12 in level 2 to training and testing the single text line datasets generated from multiple text line datasets.

### 5.2.2 CNN features and 2DBLSTM method

In the second approach, we reuse the trained CNN for feature extraction from the level 1 and the two layered 2DBLSTM network is constructed by the first 2DBLSTM layer with 64 units and the second 2DBLSTM layer with 128 units (2DBLSTM\_u:64\_u:128).

### 5.2.3 CNN based object detection method

In the third approach, we build an object detection network (Faster R-CNN) [22] using the trained parameters of the convolutional layers from the level 1 for initializing a network as an object detector and the trained parameters of the fully connected layers from the level 1 for initializing another network as a recognizer. We do not rescale the images because of the huge difference of width, height and their ratio of the images in the level 3. Thus, we employ the multi-scale regions of interesting boxes whose areas are  $4^2$ ,  $8^2$ ,  $16^2$ ,  $32^2$ ,  $64^2$ ,  $128^2$  and width-height ratios are 1:8, 1:4, 1:2, 1:1, 2:1, 4:1, 8:1.

During the training process, we use the stochastic gradient descent algorithm with the learning rate of 0.001 and the early stop scheme after 20 epochs without improvement on the validation set. When evaluating on the test set, we employ a post processing step to eliminate the overlapping bounding boxes from the prediction of FasterR-CNN. In case the ratio of intersection over union of the bounding boxes is greater than the threshold of 0.5, these bounding boxes are considered as overlapping. Among the overlapping bounding boxes, the bounding box with highest recognition score is kept and other bounding boxes are eliminated.

## 5.3 Experiments for level 3

Table 4 shows the recognition error rates of the three approaches for the level 3. Among them, the segmentation and DCRN-o\_12 combined method achieves the best result whose LER is 26.70%. However, the best SER is 82.57% by the same method. It could be decreased by linguistic context as done by many handwriting recognizers.

## 6 CONCLUSION

This paper compared several Deep Neural Network architectures to recognize anomalously deformed Kana in Japanese historical documents according to three levels contested by IEICE PRMU 2017. We achieved the single character recognition accuracy of 96.8% by the ensemble of five CNNs. For the vertical Kana text line recognition, the deep convolutional recurrent network with the stride of 12 (DCRN-o\_12) achieved

Table 4: Recognition error rates (%) on test set of level 3.

Networks	LER		SER	
	Valid set	Test set	Valid set	Test set
Segmentation + DCRN-o_12	11.72	<b>26.70</b>	49.14	<b>82.57</b>
VGGNet_c:16 + 2DBLSTM_u:64_u:128	14.45	44.18	55.82	97.16
Faster R-CNN	20.25	60.30	73.33	99.85

the best label error rate (LER) of 12.88% and the sequence error rate (SER) of 31.6%. For the unrestricted Kana recognition, the combination of vertical line segmentation and multiple line concatenation before DCRN-o\_12 achieved the best LER of 26.7% and the SER of 82.57%. The sequence error rate is very high so that linguistic context must be incorporated.

## ACKNOWLEDGMENTS

We thank Prof. Kitamoto and his group at National Institute of Informatics in Japan and the committee members of the IEICE PRMU contest for preparing the datasets and leading this contest.

## REFERENCES

- [1] S. Nicolas, T. Paquet and L. Heutte, "Enriching historical manuscripts: the Bovary project," Document Analysis Systems VI, pp.135-146, 2004.
- [2] M.S. Kim, M.D. Jang, H.I. Choi, T.H. Rhee, J.H. Kim and H.K. Kwag, "Digitalizing scheme of handwritten Hanja historical documents," Proc. 1st Int'l Workshop on Document Image Analysis for Libraries, USA, pp. 321-327, 2004.
- [3] B. Gatos, K. Ntzios, I. Pratikakis, S. Petridis, T. Konidakis and S. J. Perantonis, "An efficient segmentation-free approach to assist old Greek handwritten manuscript OCR," Pattern analysis and applications, 8 (4), pp.305-320, 2006.
- [4] T. M. Rath and R. Manmatha, "Word spotting for historical documents," IJDAR, 9 (2), pp.139-152, 2007.
- [5] A. Kitadai, J. Takakura, M. Ishikawa and M. Nakagawa, "Document image retrieval to support reading Mookans," Proc. DAS 2008, pp. 533-538, 2008.
- [6] K. Terasawa, T. Shima and T. Kawashima, "A fast appearance-based full-text search method for historical newspaper images," Proc. ICDAR 2011, Beijing, pp.1379-1383, 2011.
- [7] A. Fischer, H. Bunke, N. Naji, J. Savoy, M. Baechler and R. Ingold, "The HisDoc project. automatic analysis, recognition, and retrieval of handwritten historical documents for digital libraries," Proc. InterNational and InterDisciplinary Aspects of Scholarly Editing, pp.81-96, 2012.
- [8] C. Papadopoulos, S. Platschacher, C. Clausner and A. Antonacopoulos, "The IMPACT dataset of historical document images," Proc. HIP, pp.123-130, 2013.
- [9] T. V. Phan, K. C. Nguyen and M. Nakagawa, "A Nom historical document recognition system for digital archiving," IJDAR, 19 (1), pp.49-64, 2016.
- [10] <https://sites.google.com/view/alcon2017prmu>
- [11] <https://sites.google.com/view/alcon2017prmu/課題内容>
- [12] <http://codh.rois.ac.jp/char-shape/>
- [13] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979. doi: 10.1109/TSMC.1979.4310076
- [14] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Proc. of the 25th International Conference on Neural Information Processing Systems (NIPS'12), pp.1097-1105, 2012.
- [15] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014. CoRR: <http://arxiv.org/abs/1409.1556>
- [16] K. He, X. Zhang and S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", 2015. CoRR: <https://arxiv.org/abs/1512.03385>
- [17] R. K. Srivastava, K. Greff and J. Schmidhuber, "Highway Networks," 2015. CoRR: <https://arxiv.org/abs/1505.00387>
- [18] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," Proc. NIPS2008, Vancouver, Canada, pp. 545-552, 2008.
- [19] R. Messina and J. Louradour, "Segmentation-free handwritten Chinese text recognition with lstm-rnn," Proc. ICDAR 2015, pp. 171-175, 2015.
- [20] D. Suryani, P. Doetsch and H. Neys, "On the Benefits of Convolutional Neural Network Combinations in Offline Handwriting Recognition," Proc. ICFHR2016, China, pp. 193-198, 2016.
- [21] N. T. Ly, C. T. Nguyen, K. C. Nguyen and M. Nakagawa, "Deep Convolutional Recurrent Network for Segmentation-free Offline Handwritten Japanese Text Recognition", Proc. MOCR2017, 2017.
- [22] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015. CoRR: <https://arxiv.org/abs/1506.01497>
- [23] K. C. Nguyen and M. Nakagawa, "Enhanced Character Segmentation for Format-Free Japanese Text Recognition," Proc. ICFHR 2016, China, pp.138-143, 2016.