深層ニューラルネットワークによる手書きテキスト認識

Handwritten Text Recognition by Deep Neural Networks

By

LY TUAN NAM

A thesis submitted in fulfillment
Of the requirements for the

DEGREE OF Doctor of Philosophy OF COMPUTER SCIENCE

At the

TOKYO UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

Under supervision of **Prof. Masaki Nakagawa and Prof. Keiichi Kaneko**

Acknowledgments

First, I would like to thank my supervisors **Prof. Masaki Nakagawa and Prof. Keiichi Kaneko** of the Department of Computer and Information Sciences at Tokyo University of Agriculture and Technology. The door to their office was always open whenever I ran into a trouble spot or had a question about my research or writing. They consistently allowed this paper to be my own work but steered me in the right the direction whenever he thought I needed it. Moreover, I would like to thank other professors at the Department of Computer and Information Sciences: **Prof. Takafumi Saito**, **Prof. Ikuko Shimizu**, **Prof. Seiji Hotta**, and **Prof. Katsuhide Fujita** for their kind comments to revise this thesis and for teaching academic subjects to me for five years at TUAT. The time studying under their support is an unforgettable time in my life.

I would also like to express my deepest gratitude to the **Hirose Foundation** for financially supporting my study and thank the **iLabo Co. Ltd.** for providing data and feedback.

I thank **Prof. Asanobu Kitamoto** and his group at National Institute of Informatics in Japan and the committee members of the IEICE PRMU contest for preparing the datasets of anomalously deformed Kana and leading the PRMU contest.

I owe my thanks to all my friends and laboratory-mates; especially, **Dr. Cuong Tuan Nguyen, Dr. Hung Tuan Nguyen, and Dr. Kha Cong Nguyen** for helping me through difficult times, and for all the support they provided.

Finally, I must express my very profound gratitude to my family, especially my wonderful wife, **Trang Pham** for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author

Ly Tuan Nam

深層ニューラルネットワークによる手書きテキスト認識

Handwritten Text Recognition by Deep Neural Networks

Nam Tuan LY

Graduate School of Engineering
Tokyo University of Agriculture and Technology

Abstract

This thesis presents deep neural network-based methods for offline handwritten text recognition, and Japanese historical document recognition. Offline handwritten text recognition is still a big challenging problem due to various backgrounds, noises, diversity of writing styles, and multiple touches between characters. In this thesis, we present models of Deep Convolutional Recurrent Network (DCRN) for recognizing offline handwritten text lines without explicit segmentation of characters. The DCRN model has three parts: a feature extractor by Convolutional Neural Network (CNN); an encoder by Bidirectional Long Short-Term Memory (LSTM); and a decoder by Connectionist Temporal Classification (CTC). We also propose two upgraded version of DCRN: Attention Augmented Convolutional Recurrent Network (AACRN) model which employs 1D self-attention mechanism in the encoder, and 2D Self-Attention Convolutional Recurrent Network (2D-SACRN) which introduces a 2D self-attention mechanism in the feature extractor to help the CNN to capture the relationships between widely separated spatial regions in an input image. Since the DCRN models require a large data for training, we synthesize handwritten text line images from sentences in corpora and handwritten character patterns in the handwritten character pattern database with elastic distortions. We conducted the experiments on three public datasets: IAM Handwriting (English), Rimes (French), and TUAT Kondate (Japanese). The experimental results show that the proposed model achieves similar or better accuracy when compared to state-of-the-art models in all datasets.

For Japanese historical document recognition, we present recognition of anomalously deformed Kana sequences, 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 thesis focuses on the methods for levels 2 and 3 that won the contest. We employ the DCRN models 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. Finally, we propose an attention-based sequence to sequence model named by Attention-based Row-Column Encoder-Decoder (ARCED) for both level 2 and 3 without explicit segmentation of text lines. The experimental results prove the performances of the proposed models on the level 2 and 3 tasks.

TABLE OF CONTENTS

List of figures

List of tables

Chapter 1. Introduction

1.1. Offline Handwritten Text Recognition

Before the digital age, printed and handwritten text documents have been among the most important methods for transmitting and storing information. Nowadays, despite the abundance of electronic note-taking devices, many people still choose to write and take their notes in the traditional way with pen and paper. In addition, there are still a lot of application forms that need to be completed in handwriting with pen and paper. As a result, there is a bulk number of documents on papers that need to be processed. However, there are some drawbacks to physical handwritten documents and notes. They cannot be stored and accessed as data efficiently, making it difficult to search through them efficiently and to share them with others. In the age of digital, when computer and smart devices become popular, information is stored, processed, indexed, and searched by computer systems, making its retrieval a cheap and quick task. Handwritten documents are no exception and need to be transferred to digital format that could be easily processed by computer systems. The goal is to extract information contained in handwritten documents and to store them in a computerized format. This is the motivation of handwritten document analysis systems.

Offline handwritten text recognition is an important part of handwritten document analysis systems and has received a lot of attention from numerous researchers for decades. Starting with the recognition of isolated handwritten characters and digits, the focus has shifted to the recognition of words and sentences. Recognizing them is significantly more difficult than characters because of a large vocabulary in each language, and multiple touches between characters. Other challenging of offline handwritten text recognition are various backgrounds, noises, and diversity of writing styles. For Japanese/Chinese offline handwritten text recognition, a problem is added due to the large character set; varieties of characters mixed of thousands of Kanji characters of Chinese origin, two sets of phonetic characters, alphabets, numerals, symbols, etc.; and the difficulty of segmentation (Characters appear in a document without any word spacing).

Most early works of handwritten Japanese/Chinese text recognition were often taking the segmentation-based approach that segments or over-segments text into characters and fragments and then merges the fragments in the recognition state [1, 2]. This

segmentation-based approach is costly and error-prone, especially for cursive writing because the segmentation of characters directly affects the performance of the whole system. On the other hand, segmentation-free methods can avoid segmentation errors and have been employed for western handwritten documents based on the Hidden Markov Model (HMM) [3, 4], so far. This segmentation-free approach firstly scans the text image with a sliding window to get the sequence of images, then applies Gaussian Mixture Models (GMMs) or Neural Networks (NNs) to get the sequence of features. The sequence of features is modeled with character HMMs. A weakness of HMMs is the local modeling, which cannot capture long-term dependencies in the input sequence.

In recent years, many segmentation-free methods based on Deep Neural Networks (DNNs) and Connectionist Temporal Classification (CTC) [5–13] have been proposed and proven to be powerful models for both western and oriental text recognition. The core recognition engine has been shifted from Hidden Markov Models (HMMs) to Recurrent Neural Networks (RNNs) with CTC. The principle of the CTC-based approach is to interpret the network output as a sequence of label probabilities over all labels and use an objective function to maximize the sequence probability.

In this thesis, we present a model of Deep Convolutional Recurrent Network (DCRN) for offline handwritten Japanese text recognition without explicit segmentation of characters. The DCRN model consists of three main parts: a convolutional feature extractor using Convolutional Neural Network (CNN) to extract features from a text image; an encoder using Bi-directional Long-Short Term Memory (BLSTM) to encode the sequence of features; and a decoder using a CTC to decode the features into the final label sequence. As far as we know, this is the first approach that adopts the CTC-based model for offline handwritten Japanese text recognition.

Recently, A. Vaswani et al. [14] proposed a self-attention mechanism, which uses all position-pairs of the input sequence to extract more expressive representations for the inputs. The self-attention replaces the LSTM in both encoder and decoder and helps the sequence-to-sequence model achieve state-of-the-art results in translation via Transformer [14], speech recognition via Speech-Transformer [15], and other tasks. Based on the self-attention mechanism, we propose an upgraded version of DCRN named Attention Augmented Convolutional Recurrent Network (AACRN) which introduces 1D self-attention mechanism in the encoder. The self-attention module is complementary to RNN in the encoder and helps the encoder to capture long-range and multi-level

dependencies across an input sequence. As far as we know, this is the first approach that employs the self-attention mechanism in the encoder of the CTC-based model for offline handwritten text recognition.

Convolutional Neural Networks (CNNs) are successfully employed as feature extractors in the CTC-based models [8–13]. It processes the information in a local neighborhood, so that it might not extract information from long-distance locations in an input image. To solve this weakness of the CNN network in the feature extractor, we present a 2D Self-Attention Convolutional Recurrent Network (2D-SACRN) model with a 2D self-attention mechanism for recognizing handwritten text lines. In this model, we present a 2D self-attention mechanism in the feature extractor to help the CNN to capture the relationships between widely separated spatial regions in an input image. As far as we know, it is the first approach that employs the 2D self-attention mechanism in the feature extractor of the CTC-based model for offline handwritten text recognition. The extensive experiments on three widely used datasets: IAM Handwriting (English), Rimes (French), and TUAT Kondate (Japanese) show that the proposed model achieves similar or better accuracy when compared to state-of-the-art models in all datasets.

Deep Neural Networks, especially end-to-end models typically require a large number of patterns per category for training. However, for many handwriting datasets, especially handwritten Japanese/Chinese datasets which have many categories with over thousands, the number of patterns per category is limited, so that it is necessary to apply a data argumentation method. Many data argumentation methods for handwriting datasets have been proposed by modifying the original data such as affine transformations [12, 13], nonlinear combinations [13, 14] and Random warp grid distortion [15]. However, such method just modifies the original data, but cannot gain the real text line image. In this work, we propose a synthetic pattern generation method which synthesize handwritten text line images from sentences in corpora and handwritten character patterns in the isolated character database with elastic distortions.

1.2. Japanese Historical Document Recognition

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 a brush or wood block printed. Characters, especially Kanji of Chinese origin and Kana (a set of 46 phonetic characters made from Kanji), were deformed and cursively written, so even experts have difficulty in 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 [16–19].

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 [20]. 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. In this contest, we proposed the combination of a pre-trained CNN and an BLSTM with CTC named by Deep Convolutional Recurrent Network (DCRN) for level 2 and the DCRN combined with a vertical line segmentation method for level 3 [21]. Here, CNN stands for Convolutional Neural Network, BLSTM for Bidirectional Long Short-Term Memory Neural Network and CTC for Connectionist Temporal Classification. These methods won the best award with 12.88% character error rate (CER) for level 2 and 26.70% for level 3. After the contest, we presented their end-to-end trained versions, with the results of the new state-of-the-art accuracy of 10.90% CER for level 2 and 18.50% for level 3 [12].

This thesis is based on the previous works of the DCRN model which won the best algorithm award in the PRMU algorithm contest in 2017 but omits level 1 and focuses on level 2 and 3. Moreover, we added end-to-end-training and a two-dimensional

Bidirectional Long Short-Term Memory (2DBLSTM) based model after the contest. We compare the pretrained 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.

This thesis also introduces an attention-based sequence to sequence model named by Attention-based Row-Column Encoder-Decoder (ARCED) for both level 2 and 3 without explicit segmentation of text lines. Since Japanese historical documents were written cursively through an entire text line with neighbor text lines touching each other, a line segmentation-free approach is sought. In this model, we incorporate a row-column BLSTM in the encoder to capture the sequential order information in both the vertical and the horizontal directions and a residual LSTM network in the decoder to take advantage of entire past attention information.

1.3. Thesis organization

Chapter 2 presents surveys on two above topics: offline handwritten text recognition and Japanese historical document recognition. Next, the following chapters present my works on offline handwriting text recognition (Chapter 3), Japanese historical document recognition (Chapter 4), and Attention-based model for multiple text line recognition (Chapter 5). Finally, chapter 6 draws some conclusions and discussions about this thesis as well as shows the future works.

Chapter 2. Survey of Text Recognition

2.1. Offline Handwritten Text Recognition

Document recognition consists of two main parts: layout analysis [22] and text recognition, which we will survey briefly here.

Early works of handwritten Japanese/Chinese text recognition [1, 2] usually focused on segmentation-based methods. The segmentation-based methods firstly segment text lines into isolated characters before individually recognizing each character, but they may make over-segmentation: dividing single-character patterns into small components or under-segmentation: leaving multiple character patterns unsegmented into individual characters. Segmentation candidates and recognition candidates are combined with linguistic context and geometric context in a lattice diagram. The best path in the lattice diagram is searched to produce the recognition result [1]. Over-segmentation is preferred because split components can be merged by the best path search. However, the segmentation-based methods are costly, and the errors of this process directly affect the performance of the whole system. On the other hand, segmentation-free methods can avoid segmentation errors and have been employed for western handwritten documents using Hidden Markov Models (HMMs) [3, 4], so far. This approach firstly scans a text line image with a sliding window to get a sequence of images. Then, this sequence of images is fed into Gaussian Mixture Models (GMMs) or Neural Networks (NNs) to get a sequence of features. The sequence of features is modeled with character HMMs. Word models are obtained by concatenation of character HMMs. A weakness of HMMs is the local modeling, which cannot capture long-term dependencies in the input sequence.

In recent years, many segmentation-free methods have been proposed and shown to be effective, especially for recognizing noisy, complex, and handwritten text due to the progress of Deep Neural Networks (DNNs). They can be categorized into two main approaches: Connectionist Temporal Classification (CTC) and attention-based sequence-to-sequence methods. Both methods address the problem of variable-length between input and output in text recognition tasks.

The basic idea of CTC is to interpret an output as a probability distribution over all possible label sequences and use an objective function to maximizes the probability of the correct labeling. Early works of CTC for handwritten text recognition were presented

by A. Graves et al. [5]. They combined bidirectional Long Short-Term Memory (LSTM) and CTC to build an end-to-end trainable model for online and offline handwritten English text recognition. For offline handwritten text recognition, firstly an input image is normalized by transformation methods such as rotate and shearing transformations. Then, the feature vectors are extracted from the normalized images by a sliding window approach and fed to the BLSTM network followed by CTC to get the final sequence results. With the word accuracy of 74.1% and character accuracy of 81.8% on the IAM dataset, the BLSTM-CTC model outperformed the traditional HMM model. Following the works in [5], A. Graves et al. proposed Multidimensional Recurrent Neural Network followed by CTC for offline Arabic handwriting recognition [23]. The proposed model consists of three components: a multidimensional recurrent neural network (multidimensional LSTM in particular); a CTC output layer; and the hierarchical structure. The experiments on the IFN/ENIT database from the ICDAR 2007 Arabic handwriting recognition competition show that the proposed model outperforms all entries in the 2007 ICDAR Arabic recognition contest.

V. Pham et al. presented an end-to-end MDLSTM followed by CTC for handwritten text recognition [7]. This MDLSTM-CTC model was proposed in [23], but they have adapted the filter sizes for input images and applied dropout to the MDLSTM network. In the experiments, three handwriting datasets were used to evaluate the proposed system: Rimes, IAM and OpenHaRT containing handwritten French, English, and Arabic text, respectively. The results of the extensive experiments show that the recognition networks with dropout at the topmost layer significantly reduces the CER and WER by 10-20%, and the performance can be further improved by 30-40% if dropout is applied at multiple LSTM layers.

B. Shi et al. proposed an end-to-end trainable neural network called Convolutional Recurrent Neural Network (CRNN) for scene text recognition [8]. The CRNN model consists of three components: convolutional layers by CNN, recurrent layers by BLSTM, and a transcription layer by CTC. At the bottom of CRNN, the convolutional layers extract a feature sequence from an input image. On top of the convolutional layers, a recurrent network is built for making prediction for each frame of the feature sequence outputted by the convolutional layers. The transcription layer at the top of CRNN translates the per-frame predictions by the recurrent layers into a label sequence. The CRNN model can be jointly trained with one loss function. To evaluate the effectiveness of the proposed CRNN model, they conducted experiments on standard benchmarks for

scene text recognition including: ICDAR 2003 (IC03); ICDAR 2013 (IC13); IIIT 5k-word (IIIT5k), Street View Text (SVT), and musical score recognition. The experiment results show that CRNN achieves superior or highly competitive performance, compared with conventional methods as well as other CNN and RNN based algorithms. N. T. Ly et al. also presented the combination of pre-trained CNN and BLSTM with CTC, named Deep Convolutional Recurrent Network (DCRN) [10]. They demonstrated that the DCRN model outperforms the segmentation-based method [1] for offline handwritten Japanese text recognition. Then, they present an end-to-end version of the DCRN model for recognizing offline handwritten Japanese text [11].

Another approach, the attention-based sequence-to-sequence model has been successful applied in many tasks, such as machine translation [24, 25] and speech recognition [26]. It is also shown to be effective and achieved high accuracy in the task of text recognition [27, 28]. Following the success of MDLSTM and the attention-based sequence to sequence model, T. Bluche et al. presented the MDLSTM Attention model for handwritten paragraph recognition [29]. The MDLSTM Attention model consists of two main components: an encoder that extracts feature maps from an input image, and a sequential decoder that predicts characters from these feature maps. The author carried out the experiments on the IAM database consisting of images of handwritten English text documents. The experiment results show that the proposed model worked well on both the word and line recognition tasks.

A. Chowdhury et al. [28] proposed an attention-based model with a beam search decoder for handwritten English and French text recognition. The proposed model consists of two parts: a feature extractor that uses CNN to extract visual features from an input image, and an attention-based sequence to sequence module that maps the visual features to a sequence of characters. They also employed the form of Batch & Layer Normalization, Focal Loss, and Beam Search to improve the proposed model. The experiments on the IAM and Rimes datasets show that the proposed model provides significant boost in accuracy as compared to the standard RNN-CTC model. C. Wang et al. proposed a memory-augmented attention model for scene text recognition [27]. N. T. Ly et al. also proposed an attention-based sequence-to-sequence model with residual LSTM for recognizing multiple text lines in Japanese historical documents [30].

2.2. Historical Document Recognition

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 [31]. 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 [16]. 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. developed a system for digitizing hundreds of thousands of Nom historical documents [32]. 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) [33]. To train MQDF, pattern augmentation was applied.

For text recognition in historical documents, H. Yang et al. employed a CNN followed by CTC for Chinese text recognition in historical documents [34]. D. Valy et al. used a CNN and a 1DLSTM or 2DLSTM to recognize Khmer historical palm leaf manuscripts [35]. Ly et al. presented Deep Learning-based methods for recognizing single text line (level 2) as well as multiple text lines (level 3) in the Kana-PRMU dataset [12, 21] of Japanese historical documents as mentioned before. N. T. Ly et al. also proposed an attention-based sequence-to-sequence model with residual LSTM for recognizing multiple text lines in Japanese historical documents [30].

Chapter 3. Offline Handwritten Text Line Recognition

3.1. Introduction

In this chapter, we present models of Deep Convolutional Recurrent Network (DCRN) for recognizing offline handwritten text lines without explicit segmentation of characters. The DCRN model has three parts: a CNN feature extractor; an BLSTM encoder; and a CTC decoder. We also propose two upgraded version of DCRN: Attention Augmented Convolutional Recurrent Network (AACRN) model which employs 1D self-attention mechanism in the encoder, and 2D Self-Attention Convolutional Recurrent Network (2D-SACRN) which introduces a 2D self-attention mechanism in the feature extractor to help the CNN to capture the relationships between widely separated spatial regions in an input image. Finally, we present the Text Line Image Generation Method that synthesizes handwritten text line images from sentences in corpora and handwritten character patterns in the handwritten character pattern database with elastic distortions. The experiments are conducted on three public datasets: IAM Handwriting (English), Rimes (French), and TUAT Kondate (Japanese) to evaluate the performance of the proposed models and the effectiveness of the Text Line Image Generation Method.

The rest of this chapter is organized as follows: Section 3.2 describes the datasets. Section 3.3 presents the Deep Convolutional Recurrent Network model. Section 3.4 describes the Attention Augmented Convolutional Recurrent Network model. Section 3.5 presents the 2D Self-Attention Convolutional Recurrent Network model. Section 3.6 presents the Text Line Image Generation Method. Section 3.7 concludes the chapter.

3.2. Datasets

We conduct the experiments on the following three datasets: two widely used western handwritten datasets - IAM Handwriting [29] and Rimes [30], and one Japanese handwritten dataset - TUAT Kondate [31]. The details of them are given in the following sections.

3.2.1. TUAT Kondate.

TUAT Kondate is an online handwritten database compiled by Nakagawa Lab., Tokyo University of Agri. & Tech. (TUAT). The database stores online handwritten patterns mixed of text, figures, tables, maps, diagrams and so on. It was turned to offline patterns by thickening strokes with constant width. The handwritten Japanese portion of TUAT

Kondate comprises 13,685 text line images collected from 100 Japanese writers. We split the dataset into three subsets: 11,487 text line images collected from 84 writers for training; 800 text line images collected from 6 writers; and 1,398 text line images collected from 10 writers for testing. There are 3,345 different characters in the dataset. Figure 3.1 shows examples of images in TUAT Kondate dataset. The summary of the TUAT Kondate is given in Table 3.1.

Table 3.1. The detail of information of TUAT Kondate database.

	TUAT Kondate		
	Train set	**Valid set**	**Test set**
Number of writers	84	6	10
Number of samples	11,487	800	1,398

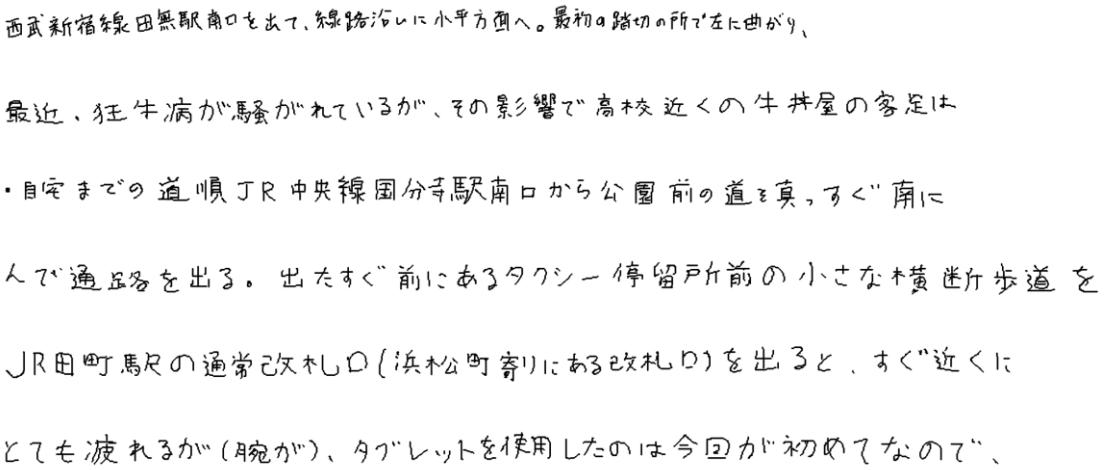


Figure 3.1. Examples from TUAT Kondate database.

We use the TUAT HANDs Nakayosi and Kuchibue handwritten Japanese character databases [16] for pretraining the weights of the CNN network of DCRN model and generating the SHTL datasets. Nakayosi contains samples of 163 writers, 10,403 character patterns covering 4,438 classes per writer. Kuchibue contains handwritten samples of 120 writers, 11,951 character patterns covering 3,345 classes per writer. The summary of the Nakayosi and Kuchibue databases are shown in Table 3.2. They are turned to offline patterns again by thickening stroke with constant width. In this work, we experimented with 3,345 classes of JIS level-1 Kanji characters (2965 classes) and kana, alpha-numerals, symbols and so on (380 classes) for pretraining the CNN model and

generating SHLT datasets. For pretraining the CNN model, we used the samples of Nakayosi for training and the samples of Kuchibue for testing. The training set (Nakayosi dataset) is randomly split into two group, with approximately 90% for training and remainder for validation.

Table 3.2. Summary of Nakayosi and Kuchibue databases.

	Nakayosi	**Kuchibue**
Number of writers	163	120
Number of classes	4,438	3,345
Number of samples	1,695,689	1,695,689

3.2.2. IAM Handwriting.

IAM Handwriting is an offline handwritten English text dataset compiled by the FKI-IAM Research Group. The dataset is composed of 13,353 text lines extracted from 1,539 pages of scanned handwritten English text, which were written by 657 different writers. All text lines in the IAM Handwriting are built using sentences provided by the Lancaster-Oslo/Bergen (LOB) corpus. We employ the IAM Aachen splits [36] shared by T. Bluche from RWTH Aachen University to split the dataset into three subsets: 6,482 lines (747 pages) for training, 2,915 (336 pages) lines for testing, and 976 lines (116 pages) for validation. There are 79 different characters in the dataset, including the space character. Figure 3.2 shows examples of images in IAM Handwriting dataset. The summary of the IAM Handwriting is given in Table 3.3.

Table 3.3. Details of the IAM Handwriting dataset.

	IAM Handwriting		
	Train set	**Valid set**	**Test set**
Text lines	6,482	976	2,915
Pages	747	116	336

3.2.3. Rimes.

Rimes is a well-known handwriting French dataset compiled by A2iA's research laboratory. The dataset consists of 11,333 lines extracted from 1,500 paragraphs for training and 778 lines extracted from 100 paragraphs for testing. The original dataset does not include the validation set, so we use the lines extracted from the last 100 paragraphs of the training set as a validation set. Consequently, the Rimes dataset consists of three

subsets: 10,532 lines for training, 801 lines for validation, and 778 lines for testing. There are 99 different characters in the dataset, including the space character. Figure 3.3 shows examples of images in IAM Handwriting dataset. The summary of the IAM Handwriting is given in Table 3.4.

Table 3.4. Details of the Rimes dataset.

	Rimes		
	Train set	**Valid set**	**Test set**
Text lines	10,532	801	778
Paragraphs	1,400	100	100

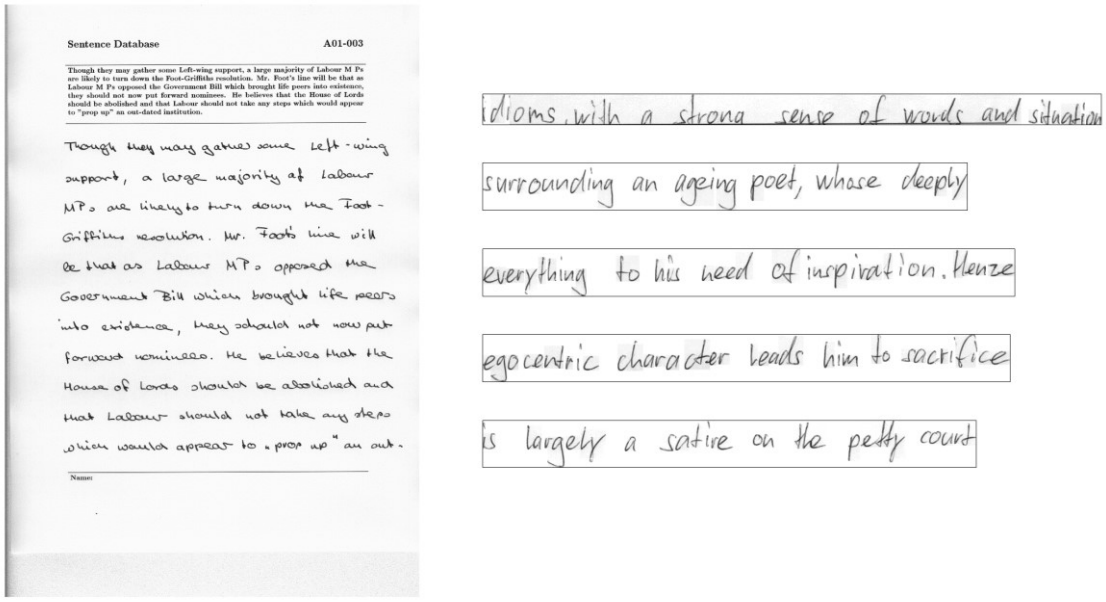
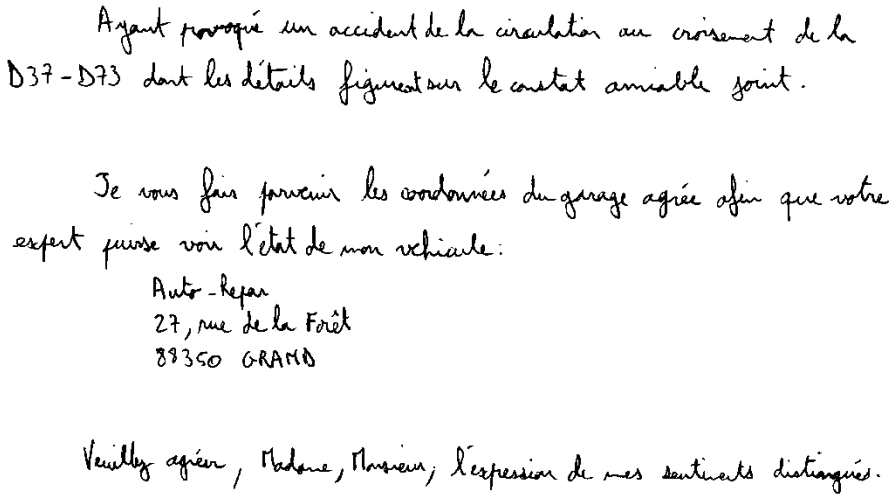


Figure 3.2. Examples from IAM Handwriting database.


Ayant provoqué un accident de la circulation au croisement de la D37-D73 dont les détails figurent sur le constat amiable joint.

Je vous fais parvenir les coordonnées du garage agréé afin que votre expert puisse voir l'état de mon véhicule :

Auto-Repar
27, rue de la Forêt
88350 GRAND

Veuillez agréer, Madame, Monsieur, l'expression de mes sentiments distingués.

Figure 3.3. Examples from Rimes database.

3.3. Deep Convolutional Recurrent Network

3.3.1. Overview of The Model

In this section, we present a model of Deep Convolutional Recurrent Network (DCRN) for recognizing offline handwritten Japanese text line images without explicit segmentation of characters. As far as we know, this is the first approach that adopts the CTC-based model for offline handwritten Japanese text recognition. The network architecture of DCRN consists of 3 components, including the convolutional feature extractor, an BLSTM encoder, and a CTC decoder, from bottom to top as shown in Figure 3.4.

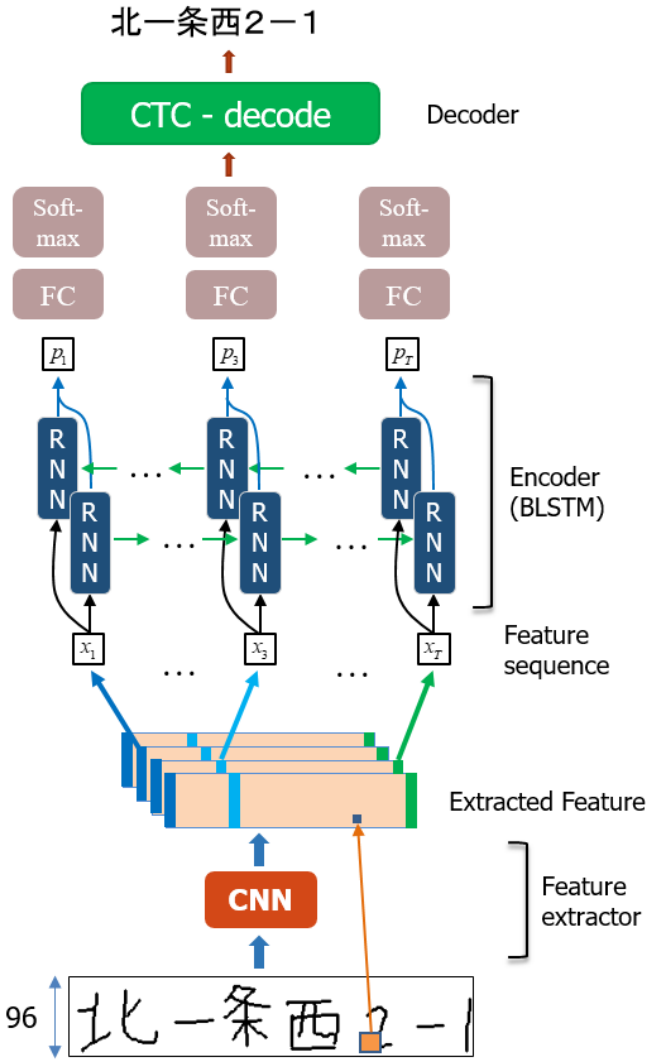


Figure 3.4. Network architecture of the DCRN model.

From the bottom of the DCRN, the convolutional feature extractor extracts a feature sequence from an input image, the encoder at the top of the convolutional feature extractor predicts each frame of the feature sequence output by the convolutional feature extractor. At the top of the DCRN, the decoder translates the pre-frame predictions by the encoder into the final label sequence.

A. Preprocessing.

Firstly, all of the text line images are scaled to the same height (the same width in Anomalously deformed Kana Sequence Recognition) of size before recognized by the DCRN model. This is necessary because in our model the feature dimension of feature sequence which extracted by the convolutional feature extractor is constant since deep BLSTM expects a fixed-size feature dimension. After resizing, in order to manage the noisy and complicated background, the text line images are converted into binary images by Otsu thresholding algorithm [17]. Figure 3.5 presents the preprocessing on anomalously deformed Kana sequence recognition.

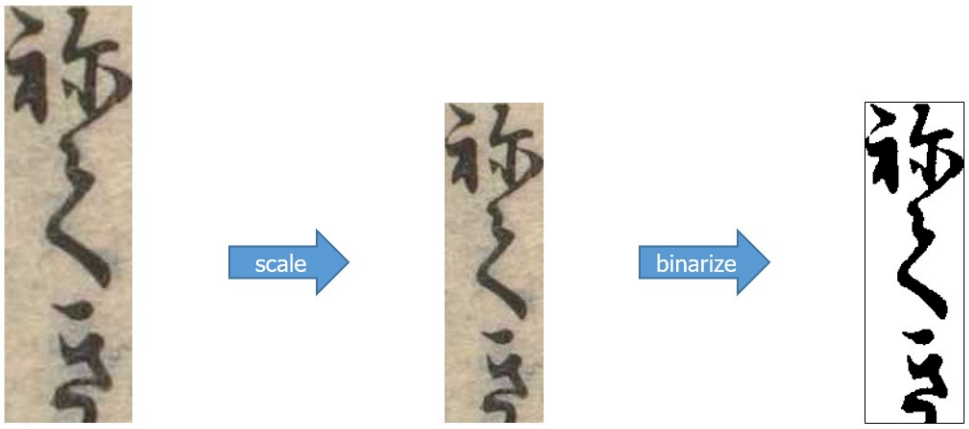


Figure 3.5. Preprocessing on anomalously deformed Kana sequence recognition.

B. Convolutional Feature Extractor.

Convolutional neural networks (CNNs) have been proven to be very powerful visual models and achieve the state-of-the-art accuracies on some tasks of computer vision such as image recognition [37] and feature representation [38, 39].

In the DCRN model, the component of convolutional feature extraction is constructed by taking the convolutional, max-pooling and full-connected layers from a standard CNN model (softmax layer are removed). Such components are used to extract a feature sequence from an input image. Before being fed into the component, all the text images

need to be scaled to the same height in order to have the same size of the input image for CNN. Then, the feature sequence is extracted from the text image by the convolutional feature extractor, which is the input of the encoder.

C. Encoder.

Recurrent neural networks (RNNs) are connectionist models containing a self-connected hidden layer. The benefits of RNNs are allowing information of previous inputs to remain in the network's internal states and the ability to make use of previous context. In the traditional RNNs, however, the vanishing gradient problem was recognized. 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 [40]. 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 to left) directions. It consists of two LSTM layers that scan the input in both the directions [6].

In DCRN model, the encoder is built on top of the convolutional feature extractor to predict a label distribution for each frame of the feature sequence extracted from the previous component. The encoder consists of the Deep Bidirectional LSTM layers which take the feature sequence from the convolutional feature extractor as the input. In the last LSTM layer, each time step of feature sequence is followed by a fully connected linear layer which converts the output feature dimension to the size of the total character set (plus 1 for CTC blank character). Finally, a softmax layer is placed at the end to generate the label probability vector at each time step.

D. Decoder.

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 π_l that produce the label sequence as shown in Eq. (3.3.1):

$$\mathrm{p}(l|x) = \sum_{\pi \in \pi_l} p(\pi|x) \tag{3.3.1}$$

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

At the top of DCRN model, the decoder decodes the pre-frame predictions made by the encoder into the final label sequence. Mathematically, decoding is to find the label sequence with the highest probability conditioned on the pre-frame predictions. To obtain the conditional probability, we employ a CTC layer as the decoder.

For decoding, we apply the CTC beam search [22] with 10 for the beam width combined with a linguistic context to obtain the final label sequence with the highest probability conditioned. In this work, we employ the tri-gram probability [23] as the linguistic context. The tri-gram probability $P(C_i|C_{i-2}, C_{i-1})$ is calculated from the corpus. It is reduced to unigram or bi-gram when C_i is the first or second character.

3.3.2. Two approaches of the DCRN model.

We have two approaches of the DCRN model: pretrained CNN approach, and End-to-End approach. The following sessions show the details of two approaches.

A. Pretrained CNN Approach.

Firstly, the CNN network is pretrained by the isolated character dataset using the stochastic gradient descent. After training the CNN network, we remove just the softmax layer or remove both the full connected layers and the softmax layer from the CNN

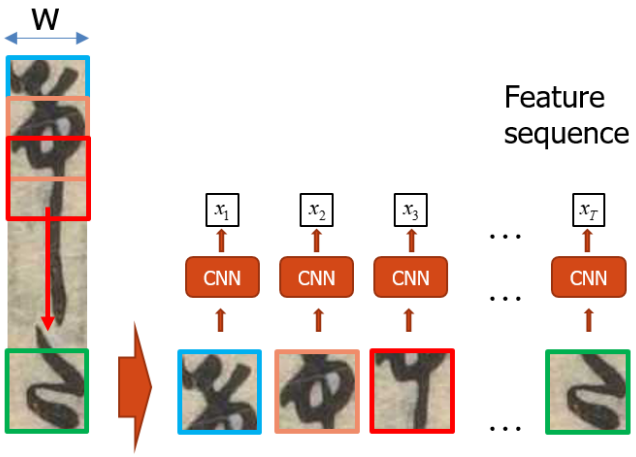


Figure 3.6. Overlapped sliding windows approach.

network to use the remaining network as the convolutional feature extractor to extract feature sequence from the input images. We call the former DCRN_o-s and the latter DCRN_o-f&s. Finally, this approach slides a sub-window of $h \times h$ (h: the height of input image in offline handwritten Japanese text recognition or the width of image in anomalously deformed Kana sequence recognition) pixels through the text line image with 12 (or 16) pixels of the sliding stride size (overlap sliding) to get an image sequence and applies the remaining CNN network to extract feature sequence from the image sequence. Figure 3.6 shows the architecture of convolutional feature extractor in this approach.

B. End-to-End Approach.

In this approach, the CNN network is constructed by taking the convolutional, max-pooling layers from a standard CNN model (fully connected and softmax layers are removed). The Leaky ReLu [21] activation is applied in all convolutional layers. Batch normalization is applied between convolutional layer and Leaky ReLu activation. We apply this CNN network to an input image of size $w \times h \times c$ (where c is the color channels of image), resulting in a multi-channel output of dimension $w' \times h' \times k$, where

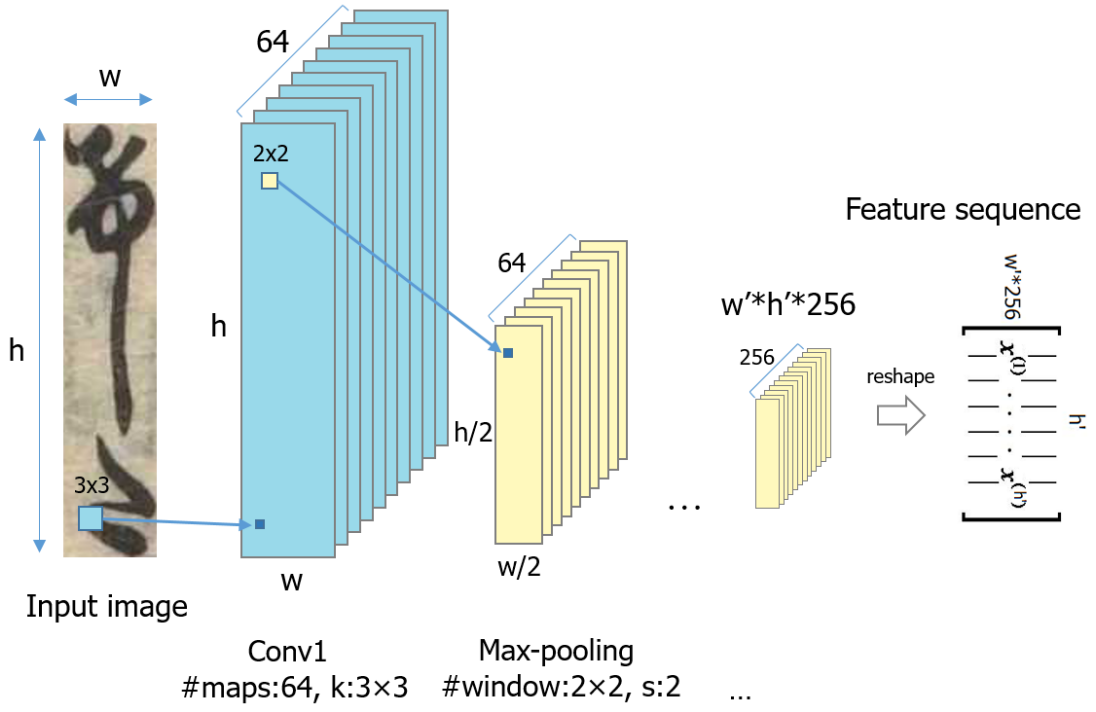


Figure 3.7. Convolutional feature extractor in the end-to-end model approach.

k is the number of feature maps in last convolutional layer, $w'$and h' depend on the w and h of input images and the amount of pooling layers in the CNN network. Then we pass the w' features of dimension $h' \times k$ to the encoder. Since the height of input images is fixed, the dimension $h' \times k$ of each feature is the same. In this approach, we do not pretrain the CNN network. However, the weights of CNN and the weights of LSTM will be end-to-end trained on the pairs of images and sequences by only one loss function. Figure 3.7 shows the architecture of convolutional feature extractor in this approach.

3.3.3. Experiments

To evaluate the performance of the proposed DCRN model, we conducted experiments on the Kondate dataset. The implementation details are described in Section A, the results of the experiments are presented in Section B and the misrecognized samples are shown in Section C.

A. Implementation Details

Pretrained CNN approach: For pretrained CNN approach, the detailed architecture of CNN network used in the convolutional feature extractor is listed in Table 3.5 in which 'k', 's', 'p' and 'group' denote to kernel size, stride, padding size and group size, respectively. It contains seven learned layers - four convolutional layers alternatively by four max-pooling layers, two full-connected layers and a softmax layer finally (3345 class). Each convolutional and fully-connected layer is followed by Maxout units [25], using the group size of 2. Firstly, the CNN network is pretrained by the TUAT Nakayosi and Kuchibue using stochastic gradient descent with a batch size of 64 samples with the learning rate of 0.01 and the momentum of 0.95 on GPU. After training the CNN network, we remove just the softmax layer or both the full connected layers and the softmax layer from the CNN network to use the remaining network as the convolutional feature extractor. We call the former DCRN_o-s and the latter DCRN_o-f&s.

At the encoder, we employ Deep BLSTM network with 256 nodes of two layers. A fully connected layer and a softmax layer with the node size equal to the character set size (n=3347) are applied after each time step of Deep BLSTM network. The encoder and the decoder are trained by using online steepest descent with the learning rate of 0.0001 and the momentum of 0.9. We use the training set of Kondate for training and the testing set of Kondate for evaluating the performance of this approach. The validation set of Kondate is used for turning hyperparameters and avoid overfitting in the DCRN model.

Table 3.5. Network configuration of the CNN model in the pretrained CNN approach.

Type	Configurations
Input	96×96 image
Conv1 - Maxout	#maps:32, k:5×5, s:1, p:0, group:2
MaxPooling1	Window:2×2, s:2
Conv2 - Maxout	#maps:32, k:3×3, s:1, p:0, group:2
MaxPooling2	Window:2×2, s:2
Conv3 - Maxout	#maps:64, k:3×3, s:1, p:0, group:2
MaxPooling3	Window:2×2, s:2
Conv4 - Maxout	#maps:64, k:5×5, s:1, p:0, group:2
MaxPooling4	Window:2×2, s:2
Full-connected - Maxout	#nodes:400, group:2
Full-connected - Maxout	#nodes:400, group:2
Softmax	#nodes: 3345(number class)

End-to-end approach: For the End-to-End approach, the architecture of CNN network used in the convolutional feature extractor is shown in Table 3.6 in which 'maps', 'k', 's' and 'p' denote the number of kernels, kernel size, stride and padding size of convolutional layers respectively. It consists of 8 convolutional layers. Batch normalization is applied after the 2nd, 4th, 6th and 8th convolutional layers followed by Max-Pooling layers. The Leaky ReLu [20] activation function is applied in all convolutional layers.

At the encoder, we employ Deep BLSTM network with 128 hidden nodes of three layers. In order to prevent overfitting when training the model, the dropout (dropout rate=0.2) is also applied in each layer of Deep BLSTM. A fully connected layer and a softmax layer with the node size equal to the character set size (n=3347) are applied after each time step of Deep BLSTM network.

The end-to-end approach is trained using stochastic gradient descent with the learning rate of 0.001 and the momentum of 0.9. The training process stops when the recognition accuracy of validation set does not gain after 10 epochs.

Table 3.6. Network configuration of the CNN model in the end-to-end model approach.

Type	Configurations
Input	96×w image
Conv1 - LReLu	#maps:32, k:3×3, s:1, p:1
Conv2 - Batch Norm - LReLu	#maps:32, k:3×3, s:1, p:1
MaxPooling1	#window:2×2, s:2
Conv3 - LReLu	#maps:64, k:3×3, s:1, p:1
Conv4 - Batch Norm - LReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling2	#window:2×2, s:2
Conv5 - LReLu	#maps:128, k:3×3, s:1, p:1
Conv6 - Batch Norm - LReLu	#maps:128, k:3×3, s:1, p:1
MaxPooling3	#window:2×2, s:2
Conv7 - LReLu	#maps:256, k:3×3, s:1, p:1
Conv8 - Batch Norm - LReLu	#maps:256, k:3×3, s:1, p:1
MaxPooling4	#window:2×2, s:2

B. Experiment Results

In order to evaluate the performance of the AARCN model, we employ the terms of Character Error Rate (CER) and Sequence Error Rate (SER) that are defined as follows:

$$\mathrm{CER}(h,S') = \frac{1}{Z}\sum_{(x,z)\in S'} \mathrm{ED}(h(x),z) \tag{3.3.2}$$

$$\mathrm{SER}(h,S') = \frac{100}{|S'|}\sum_{(x,z)\in S'} \begin{cases} 0 & \text{if } h(x)=z \\ 1 & \text{otherwise} \end{cases} \tag{3.3.3}$$

where Z is the total number of target labels in S' and ED(p, q) is the edit distance between two sequences p and q.

The first experiment evaluated the performance of the pretrained CNN approach and the end-to-end approach without using the linguistic context. Table 3.7 shows the recognition rate on the validation and test sets of Kondate. In the pretrained CNN approach, DCRN_o-s obtained LER of 6.44% and SER of 25.89% on the test set, it is

compared with the results of DCRN_o-f&s. The results imply that the DCRN_o-s model, the convolutional feature extractor made by only removing the softmax layer from the CNN model, works better than DCRN_o-f&s, the convolutional feature extractor made by removing both the fully connected layers and the softmax layer from the CNN model. End-to-End obtained LER of 3.65% and SER of 17.24% on the test set. The results imply that the end-to-end approach substantially outperforms the pretrained CNN approach.

Table 3.7. Label Error Rate (LER) and Sequence Error Rate (SER) on Kondate.

Model	**LER**		**SER**	
	Valid set	**Test set**	**Valid set**	**Test set**
DCRN_o-f&s	11.74%	6.95%	39.33%	28.04%
DCRN_o-s	11.01%	6.44%	37.38%	25.89%
End-to-End	5.22%	3.65%	24.47%	17.24%

Secondly, we evaluated the performance of the pretrained CNN approach and the end-to-end approach with the linguistic context [23]. Table 3.8 shows the recognition rate of these approaches on the test set when combined with the linguistic context. It is compared with the previous segmentation-based method with the linguistic context. The results show that both the pretrained CNN approach and the end-to-end approach are superior to the segmentation-based method and its recognition accuracy is further improved when the linguistic context is integrated.

Table 3.8. LER and SER on Kondate when combined with the linguistic context.

Model	**Test set**	
	LER	**SER**
Segmentation based [1]	11.2%	48.53%
DCRN_o-f&s	6.68%	26.97%
DCRN_o-s	6.10%	24.39%
End-to-End	3.52%	16.67%

C. Correctly recognized and misrecognized samples

There are a total of 362 misrecognized samples among 1398 samples. Most of them are missing some characters in the ground-truth. Figure 3.8 shows some misrecognized

samples by DCRN_o-s whose sequence error rate is 25.89%. For each sample, the upper image is an input handwritten text line image and the text bounded by the lower blue rectangular shows the ground-truth and the recognition result separated by "->".

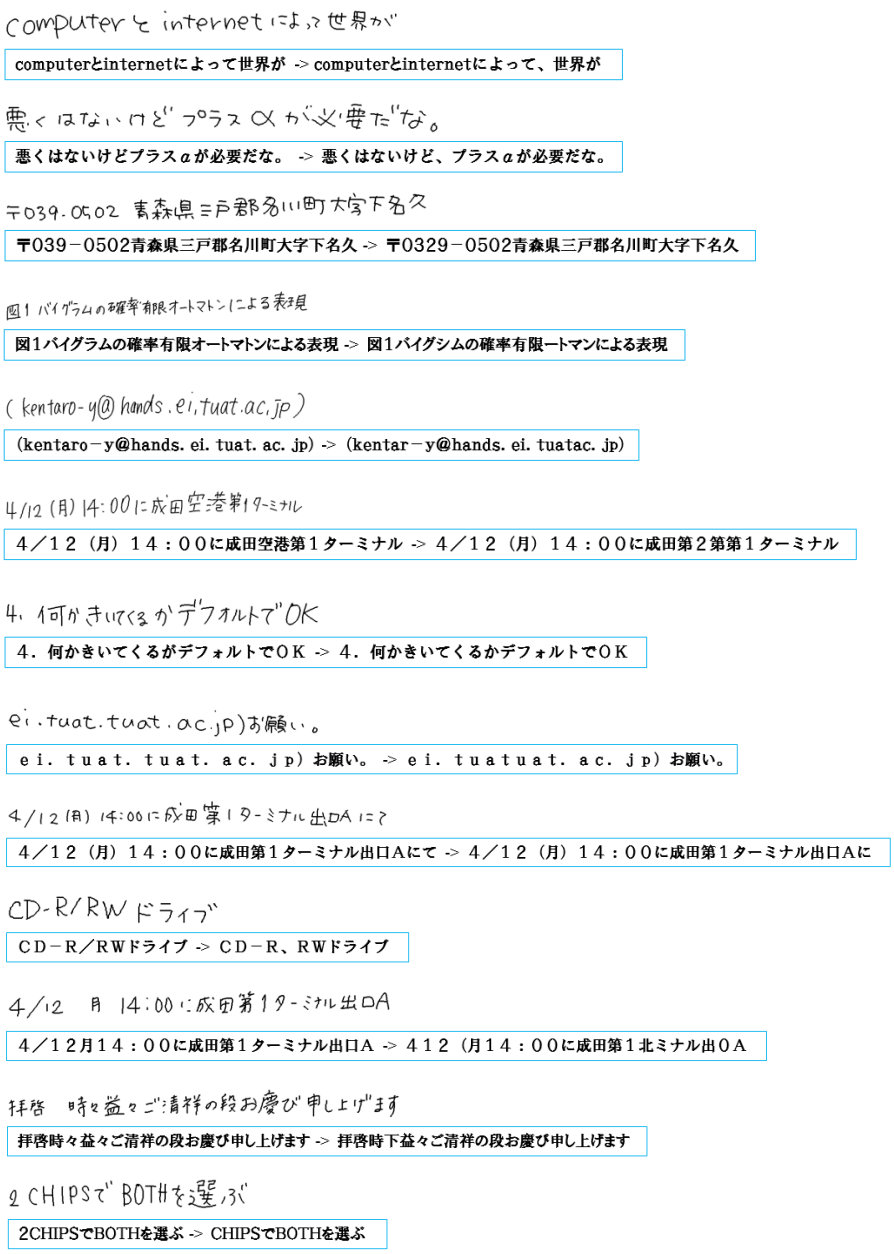


Figure 3.8. Some mispredicted samples by DCRN_o-s.

3.4. Attention Augmented Convolutional Recurrent Network

The RNNs, such as Gated recurrent unit (GRU) or Long-short term memory (LSTM), are good at sequence modeling and solve the weakness of the local modeling of HMMs. However, the number of hidden nodes in RNNs is usually fixed, which implies all historical information is compressed into a fixed-length vector, so that RNNs are difficult to capture long-range context. Recently, A. Vaswani et al. [14] proposed a self-attention mechanism in the Transformer model, which achieved the state-of-the-art performance in some machine translation tasks. The self-attention mechanism can capture the dependencies between different positions of arbitrary distance in an input sequence and replaces the LSTM in both the encoder and the decoder of the sequence-to-sequence models. Based on the self-attention mechanism, we propose an upgraded version of DCRN named Attention Augmented Convolutional Recurrent Network (AACRN) for recognizing handwritten Japanese text lines. In this model, we employ the self-attention mechanism in the encoder to help the encoder to capture long-range and multi-level dependencies across an input sequence. As far as we know, this is the first approach that employs the self-attention mechanism in the encoder of the CTC-based model for offline handwritten text recognition.

3.4.1. Self-Attention Mechanism

Self-attention is a mechanism that uses all position-pairs of the input sequence to extract more expressive representations for the inputs. Therefore, the self-attention mechanism can capture the dependencies between different positions of arbitrarily distance in the

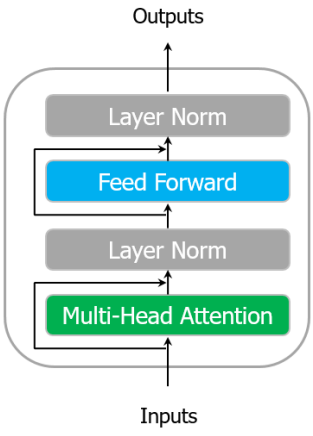


Figure 3.9. Self-attention layer.

input sequence. The self-attention layer [14] consists of two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. A residual connection [41] followed by layer normalization is applied after each of the two sub-layers. Figure 3.9 shows the architecture of the self-attention layer. Next, we will describe how the self-attention layer works. Let $X \in R^{T \times d_x}$ denote an input to the self-attention layer. The first sub-layer performs multi-head attention to the input *X*. Each head *i* of the n_h heads compute the *queries* Q^i, *keys* K^i, and *values* V^i from *X* by linear projections and then performs Scaled Dot-Product Attention to the *queries*, *keys*, and *values* to compute the output as shown in Eq. (3.4.1) and Eq. (3.4.2):

$$Q_i = XW_i^Q \quad K_i = XW_i^K \quad V_i = XW_i^V \tag{3.4.1}$$

$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d_k}}\right) V_i \tag{3.4.2}$$

where the projections are parameter matrices $W_i^Q, W_i^K \in \mathbb{R}^{d_x \times d_k}$ and $W_i^V \in \mathbb{R}^{d_x \times d_x/n_h}$. All heads are concatenated and again projected to give the output of the multi-head sub-layer, as shown in Eq. (3.4.3):

$$\text{O}_{\text{MultiHead}} = \text{Concat}\left(\text{head}_1, ..., \text{head}_{n_h}\right) W^O \tag{3.4.3}$$

where the projection is a parameter matric $W_i^O \in \mathbb{R}^{d_x \times d_x}$. The output and the input of the multi-head sub-layer are fed to the layer normalization and to the second sub-layer: a position-wise fully connected feed-forward network. Finally, the output of the self-attention layer is computed as shown in Eq. (3.4.4) and Eq. (3.4.5):

$$X_2 = \text{LN}\left(\text{O}_{\text{MultiHead}} + X\right) \tag{3.4.4}$$

$$O_{\text{SelfAttnLayer}} = \text{LN}\left(\text{FFN}\left(X_2\right) + X_2\right) \tag{3.4.5}$$

Where LN and FFN are the layer normalization and the position-wise fully connected feed-forward network, respectively.

3.4.2. Overview of The Model

The AACRN model consists of three main parts: a convolutional feature extractor, a self-attention-based encoder, and a CTC decoder, from bottom to top, as shown in Figure 3.10 They are described in the following sections.

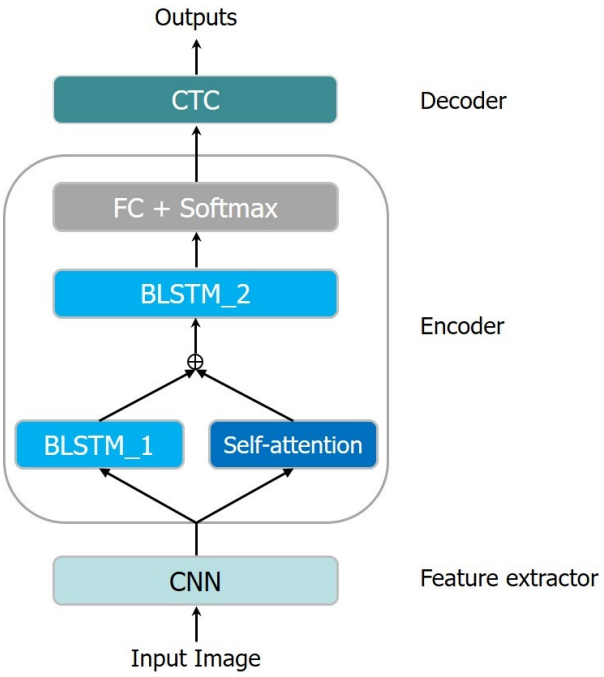


Figure 3.10. Network architecture of the AACRN model.

A. Feature extractor

Similar to the DCRN model, we employ a standard CNN network without fully connected layers to build the feature extractor in the AACRN model. This CNN network is constructed by taking the convolutional, max-pooling layers from a standard CNN model to extract a sequence of features from an input image. (fully connected layers are removed). All the images will be scaled to the same height before fed into the network. As shown in Figure 3.11, given an input image of size $w \times h \times c$ (where c is the color

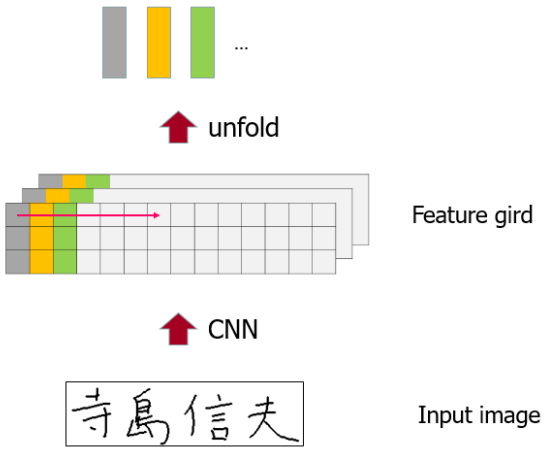


Figure 3.11. Feature extraction for an input image.

channels of image), the CNN network extracts a feature gird F of size $w' \times h' \times k$, where k is the number of feature maps in the last convolutional layer, $w'$and h' depend on the w and h of input images and the number of pooling layers in the CNN network. Then, the feature gird F will be unfolded to a sequence of features column by column from left to right in each feature map. The sequence of features will be fed to the encoder.

B. Self-attention based encoder

In the AACRN model, we use the combination of the self-attention layers and the BLSTM network to build the encoder, which encodes a sequence of features extracted from the previous component into a sequence of label probability vectors. The self-attention layers help the encoder capture the dependencies between different positions with arbitrarily distance in the input. Meanwhile, the BLSTM network helps the encoder focus on the dependencies of nearby positions. The encoder consists of three main parts: a self-attention block, which consists of several self-attention layers, two BLSTM networks which denote BLSTM_1 and BLSTM_2, and a fully connected layer, as shown in Figure 3.4.2. The output of the self-attention block and BLSTM_1 will be concatenated before being fed into BLSTM_2. Then, the output of BLSTM_2 will be fed into the fully connected layer, which converts the output feature dimension to the size of the total character set (plus 1 for CTC blank character). Finally, a softmax layer is placed at the end to generate the label probability vector at each time step.

Let $F = (f_1, f_2 \cdots f_n)$, $E = (e_1, e_2 \cdots e_n)$ and $H = (h_1, h_2 \cdots h_n)$ denote the sequence of features, the sequence of label probability vectors, and the output of the combination of the self-attention block and BLSTM_1, respectively, where n is the number of feature vectors. Then, we have:

$$H = \text{Concat}\left(\text{BLSTM\_1}\left(F\right), \text{SelfAttn}\left(F\right)\right) \tag{3.4.6}$$

$$E = \text{Softmax}\left(\text{FC}\left(\text{BLSTM\_2}\left(H\right)\right)\right) \tag{3.4.7}$$

C. Decoder

At the top of the AACRN model, the decoder generates the final label sequence from the sequence of label probability vectors made by the encoder. Mathematically, the decoder finds the label sequence with the highest probability conditioned on the sequence of the label probability vectors. To obtain the conditional probability, we employ the CTC algorithm [42] as the decoder.

The whole model can be trained end-to-end by the CTC loss and stochastic gradient descent algorithms. For decoding, we apply the CTC beam search with the beamwidth of 10 to obtain the final label sequence with the highest probability conditioned.

3.4.3. Experiments

To evaluate the performance of the proposed AACRN model, we conducted experiments on the Kondate Japanese text line dataset. The implementation details are described in Sec A; the results of the experiments are presented in Sec. B and the correctly recognized and misrecognized samples are shown in Sec. C.

A. Implementation Details

The architecture of the CNN network in the feature extractor is shown in Table 3.9, where 'maps', 'k', 's' and 'p' denote the number of kernels, kernel size, stride and padding size of convolutional layers, respectively. It consists of six convolutional (Conv) layers. After all Conv layers, batch normalization [43] followed by the ReLU activation function is applied in order to normalize the inputs to the nonlinear activation. Each Conv layer in the first five Conv layers is followed by Max-Pooling layers.

Table 3.9. Network configuration of the CNN model.

Type	Configurations
Input	h×w image
Conv1 - Batch Norm - ReLu	#maps:32, k:3×3, s:1, p:1
MaxPooling1	#window:2×2, s:2×2
Conv2 - Batch Norm - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling2	#window:2×2, s:2×2
Conv3 - Batch Norm - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling3	#window:2×2, s:2×2
Conv4 - Batch Norm - ReLu	#maps:128, k:3×3, s:1, p:1
MaxPooling4	#window:1×2, s:1×2
Conv5 - Batch Norm - ReLu	#maps:256, k:3×3, s:1, p:1
MaxPooling5	#window:2×1, s:2×1
Conv6 - Batch Norm - ReLu	#maps:256, k:3×3, s:1, p:1

At the encoder, the self-attention block consists of six self-attention layers where each self-attention layer is composed of eight heads and 2048 nodes of one fully connected layer. Both BLSTM_1 and BLSTM_2 is composed of forward and backward layers where each forward or backward layer is one LSTM layer having 256 hidden nodes. In order to prevent overfitting when training the model, the dropout (dropout rate=0.2) is also applied in each layer of the two bidirectional BLSTM networks. A fully connected layer and a softmax layer with the node size equal to the character set size (n=3347 for Kondate dataset) are applied at the end of the encoder.

The AACRN model is trained using stochastic gradient descent with a learning rate of 0.001 and a momentum of 0.9. The training process stops when the recognition accuracy of the validation set does not gain after ten epochs.

B. Experiment Results

In order to evaluate the performance of the AARCN model, we employ the terms of Character Error Rate (CER) and Sequence Error Rate (SER) that are defined in Eq. (3.3.2) and Eq. (3.3.3).

Japanese text recognition

The first experiment evaluated the performance of the AACRN model on the Kondate - offline handwritten Japanese text line dataset. Table 3.10 compares the recognition error rates by the AACRN model and the previous works of the DCRN models on the test set of the Kondate dataset.

Table 3.10. Recognition error rates (%) on the test set of Kondate dataset.

Model	Kondate	
	CER	SER
DCRN_o-f&s	6.95	28.04
DCRN_o-s	6.44	25.89
End-to-End DCRN	3.65	17.24
AACRN	2.73	15.74

The AACRN model obtained CER of 2.73% on the test set of Kondate dataset, respectively. The results imply that the AACRN model substantially outperforms both

pretrained approach and End-to-End approach of the DCRN model on the test sets of Kondate dataset.

Effects of self-attention block

To verify the effect of the self-attention block in the encoder, we prepared one variation, which was the same as the AACRN model except using the self-attention block in the encoder. This model is named AACRN_w/o_selfAttn. Table 3.11 compares its recognition error rates with the AACRN model on the test set of the Kondate dataset.

Table 3.11. Recognition error rates (%) with different encoders.

Model	Kondate	
	CER	SER
AACRN_w/o_selfAttn	3.44	19.67
AACRN	2.73	15.74

In the test sets of the Kondate dataset, the AACRN model outperforms the AACRN_w/o_selfAttn model. The results show that the self-attention block in the encoder improves the performance of the AACRN model for the Japanese text recognition tasks. This seems to be due to the self-attention block that helps the encoder capture the dependencies between different positions in the input sequence.

3.5. 2D Self-Attention Convolutional Recurrent Network

Convolutional Neural Networks (CNNs) are successfully employed as feature extractors in the CTC-based models [8–13]. It processes the information in a local neighborhood, so that it might not extract information from long-distance locations in an input image. To solve this weakness of the CNN network in the feature extractor of DCRN, we propose an upgraded version of DCRN named 2D Self-Attention Convolutional Recurrent Network (2D-SACRN) model for offline handwritten text line recognition. In this model, we present a 2D self-attention mechanism in the feature extractor to help the CNN to capture the relationships between widely separated spatial regions in an input image. As far as we know, it is the first approach that employs the 2D self-attention mechanism in the feature extractor of the CTC-based model for offline handwritten text recognition.

3.5.1. 2D Self-Attention Mechanism.

Convolutional Neural Networks have been proven to be compelling models and achieve state-of-the-art results in many computer vision tasks. However, convolutional operation processes the information in a local neighborhood. Thus, it is difficult to obtain information from long-distance locations. X. Wang et al. proposed the non-local operations in Non-local Neural Networks for capturing long-range dependencies in an image or video [44]. H. Zhang et al. adapted the non-local model in [44] to introduce self-

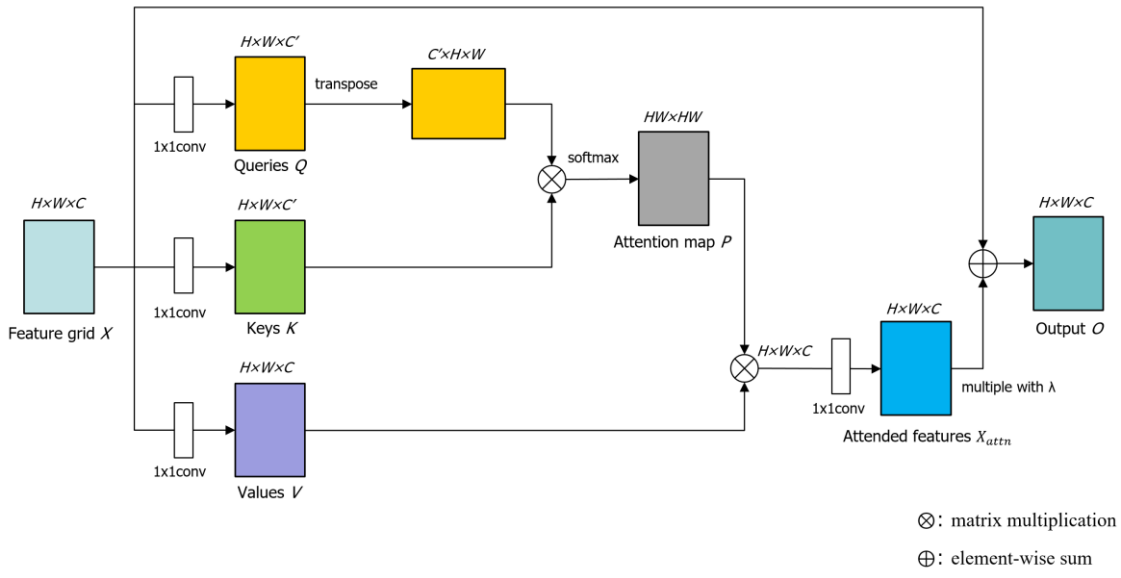


Figure 3.12. Architecture of 2D Self-Attention block.

attention to the GAN framework, helping both the generator and the discriminator capture the relationships between widely separated spatial regions [45]. Based on their works, in this paper, we present a 2D self-attention block in the feature extractor to help it capture the relationships between widely separated spatial regions in an input image. The architecture of the 2D self-attention block is shown in Figure 3.12.

Let $X \in R^{H \times W \times C}$ denote a feature grid input to the 2D self-attention block (where *H*, *W*, and *C* are *height*, *width,* and the number of channels of the feature grid *X*, respectively). Firstly, the 2D self-attention block transforms the feature grid *X* into three feature grids: queries $Q \in R^{H \times W \times C'}$, keys $K \in R^{H \times W \times C'}$, and values $V \in R^{H \times W \times C}$ by linear projections as shown in Eq. (3.5.1):

$$Q = X \otimes W_Q \quad K = X \otimes W_K \quad V = X \otimes W_V \tag{3.5.1}$$

where the projections are the learned parameter matrices $W_Q \in \mathbb{R}^{C \times C'}$, $W_K \in \mathbb{R}^{C \times C'}$, and $W_V \in \mathbb{R}^{C \times C}$ with each implemented as a 1×1 convolution layer.

The 2D self-attention maps $P \in R^{H \times W \times H \times W}$ are calculated from the queries *Q* and the keys *K* as shown in Eq. (3.5.2) and Eq. (3.5.3):

$$s_{ijqk} = K_{ij} \otimes Q_{qk}^T \tag{3.5.2}$$

$$P_{ijqk} = \frac{\exp\left(S_{ijqk}\right)}{\sum_{q=0,k=0}^{q=H,k=W} \exp\left(S_{ijqk}\right)} \tag{3.5.3}$$

where Q^T is the transpose of the queries *Q* and P_{ijqk} indicates how the ij^{th} location in the feature grid *X* attend to the qk^{th} location in the feature grid *X*.

Then, the attended feature grid X_{attn} are computed from the 2D self-attention maps *P* and the values *V* as shown in Eq. (3.5.4):

$$X_{attn} = (P \otimes V) \otimes W_F \tag{3.5.4}$$

where $W_F \in \mathbb{R}^{C \times C}$ is the learned parameter matrices implemented as a 1×1 convolution layer.

Finally, the output of the 2D self-attention block is calculated from the attended features X_{attn}, and the input feature grid X as follow:

$$O = X_{attn} * \lambda + X \tag{3.5.5}$$

where λ is a learnable scalar, and it is initialized as 0.

3.5.2. Overview of The Model

The 2D-SACRN model is composed of three main components: a feature extractor, a recurrent encoder, and a CTC-decoder, as shown in Figure 3.13 They are described in the following sections.

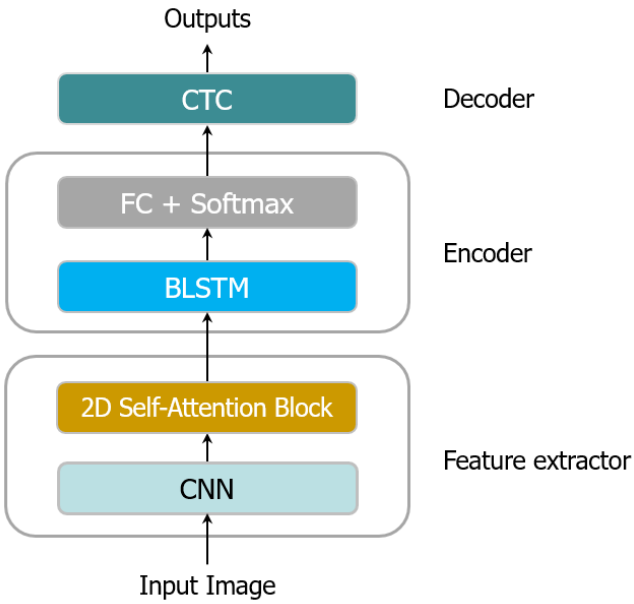


Figure 3.13. Network architecture of the 2D-SACRN model.

A. Feature Extractor

In the 2D-SACRN model, we employ a CNN network followed by a 2D self-attention block to build the feature extractor. The CNN network is constructed by taking the convolutional and max-pooling layers from a standard CNN network while removing fully connected, and Softmax layers. Given an input image of size $w \times h \times c$ (where c is the color channels of image), the CNN network extracts a feature gird F of size $w' \times h' \times k$, where k is the number of feature maps in the last convolutional layer, and $w'$and h' depend on the w and h of input images and the number of pooling layers in the CNN

network. Then, the feature grid F is fed into the 2D self-attention block to get the final attended feature grid F_{attn}. Finally, the final attended feature grid F_{attn} is unfolded to a feature sequence column by column from left to right in each feature map. The feature sequence is fed into the encoder.

B. Encoder

At the top of the feature extractor, the encoder encodes the feature sequence extracted from the feature extractor into a sequence of label probabilities. Mathematically, the encoder predicts label-probabilities for each feature in the feature sequence. In the 2D-SACRN model, we use a BLSTM network followed by a fully connected layer and a Softmax layer to build the encoder. The BLSTM network takes the feature sequence from the feature extractor as the input. Then, the output of the BLSTM network is fed into the fully connected layer, which converts the output feature dimension to the size of the total character set (plus 1 for CTC blank character). Finally, the Softmax layer, which is placed at the end of the encoder, generates the label probabilities at each time step.

Let $F_{seq} = (f_1, f_2 \cdots f_n)$, $E = (e_1, e_2 \cdots e_n)$ and $H = (h_1, h_2 \cdots h_n)$ denote the feature sequence, the sequence of label probabilities, and the output of the BLSTM network, respectively, where n is the number of feature vectors. Then, we have:

$$H = \mathrm{BLSTM}\left(F_{seq}\right) \tag{3.5.6}$$

$$E = \mathrm{Softmax}\left(\mathrm{FC}\left(H\right)\right) \tag{3.5.7}$$

C. Decoder

At the top of the 2D-SACRN model, the decoder converts the sequence of label probabilities made by the encoder into a final label sequence. Mathematically, the decoding process is to find the final label sequence with the highest probability conditioned on the sequence of label probabilities. CTC [42] is a specific loss function designed for sequence labeling tasks where it is difficult to segment the input sequence into the final segmented sequence that exactly matches a target sequence. In this work, we employ the CTC algorithm to build the decoder to obtain the conditional probability.

The whole system is trained end-to-end using stochastic gradient descent algorithms to minimize the CTC loss. For the decoding process in the testing phase, we apply the CTC

beam search with the *beamwidth* of *2* to obtain the final label sequence with the highest probability conditioned.

3.5.3. Experiments

To evaluate the performance of the proposed 2D-SACRN model, we conducted experiments on the three datasets: IAM handwriting, Rimes, and TUAT Kondate. The implementation details are described in Sec A; the results of the experiments are presented in Sec. B; and the visualization of the 2D self-attention map is shown in Sec. C.

A. Implementation Details

IAM and Rimes datasets. In the experiments on the two western datasets, the architecture of the CNN network in the feature extractor is ConvNet-1 as shown in Table 3.12, where 'maps', 'k', 's' and 'p' denote the number of kernels, kernel size, stride and padding size of convolutional layers, respectively. It consists of five convolutional (Conv) blocks. Each Conv block consists of one Conv layer followed by the Batch normalization [43] and the ReLU activation. To reduce overfitting, we apply dropout at the input of the last three Conv blocks (with dropout probability equal to 0.2).

Table 3.12. Network configurations of the CNN in the feature extractor.

Type	**Configurations**	
	ConvNet-1	**ConvNet-2**
Input	h×w image	h×w image
Conv Block 1	#maps:16, k:3×3, s:1, p:1	#maps:16, k:3×3, s:1, p:1
Max-Pooling1	#window:2×2, s:2×2	#window:2×2, s:2×2
Conv Block 2	#maps:32, k:3×3, s:1, p:1	#maps:32, k:3×3, s:1, p:1
Max-Pooling2	#window:2×2, s:2×2	#window:2×2, s:2×2
Conv Block 3	#maps:48, k:3×3, s:1, p:1	#maps:48, k:3×3, s:1, p:1
Max-Pooling3	#window:1×2, s:1×2	#window:2×2, s:2×2
Conv Block 4	#maps:64, k:3×3, s:1, p:1	#maps:64, k:3×3, s:1, p:1
Max-Pooling4	#window:2×1, s:2×1	#window:1×2, s:1×2
Conv Block 5	#maps:80, k:3×3, s:1, p:1	#maps:80, k:3×3, s:1, p:1
Max-Pooling5		#window:2×1, s:2×1
Conv Block 6		#maps:128, k:3×3, s:1, p:1

At the encoder, we employ a Deep BLSTM network with 256 hidden nodes of five layers. To prevent overfitting when training the model, the dropout (dropout rate=0.5) is also applied in each layer of the Deep BLSTM network. A fully connected layer and a softmax layer with the node size equal to the character set size (n=80 for IAM and 100 for Rimes) are applied after each time step of the Deep BLSTM network.

TUAT Kondate dataset. In our experiments on the TUAT Kondate dataset, the architecture of the CNN network in the feature extractor is ConvNet-2, which consists of six Conv blocks, as shown in Table 3.12. The Deep BLSTM network in the encoder has three BLSTM layers with 256 hidden nodes of each layer. The other configurations are the same as the 2D-SACRN model in the experiments on the two western datasets.

B. Experiment Results

In order to evaluate the performance of the 2D-SACRN model, we employ the terms of Character Error Rate (CER), Word Error Rate (WER), and Sequence Error Rate (SER) that are defined as follows:

$$\mathrm{CER}(h,S') = \frac{1}{Z} \sum_{(x,z)\in S'} \mathrm{ED}(h(x),z) \tag{4.5.8}$$

$$\mathrm{WER}(h,S') = \frac{1}{Z_{word}} \sum_{(x,z)\in S'} \mathrm{ED}_{word}(h(x),z) \tag{4.5.9}$$

$$\mathrm{SER}(h,S') = \frac{100}{|S'|} \sum_{(x,z)\in S'} \begin{cases} 0 & \text{if } h(x)=z \\ 1 & \text{otherwise} \end{cases} \tag{4.5.10}$$

where Z is the total number of target labels in S' and $ED(p, q)$ is the edit distance between two sequences p and q, while Z_{word} is the total number of words in S' and $ED_{word}(p, q)$ is the word-level edit distance between two sequences p and q.

English and French Text Recognition:

The first experiment evaluated the performance of the 2D-SACRN model on the two western handwritten datasets: IAM Handwriting and Rimes in terms of CER and WER. To fairly compare with the previous models, we do not use any data augmentation techniques as well as linguistic context information. Table 3.13 shows the recognition error rates by the 2D-SACRN model and the previous models [7, 9, 13, 46–50] on the test set of IAM Handwriting and Rimes datasets without using the language model.

On the IAM Handwriting dataset, the 2D-SACRN model achieved CER of 6.76% and WER of 20.89%. These results show that the 2D-SACRN model achieves the state-of-the-art accuracy and outperforms the best model in [50] by about 8% of CER and 15% of WER on the IAM Handwriting dataset. On the Rimes dataset, the 2D-SACRN model achieved CER of 3.43% and WER of 11.92%. Although its CER was considerably larger than the current state-of-the-art [48], its WER was the best.

From the above results, we conclude that the 2D-SACRN model achieves similar or better accuracy when compared to the state-of-the-art models in both IAM Handwriting and Rimes datasets.

Table 3.13. Recognition error rates (%) on IAM and Rimes datasets.

Model	**IAM**		**Rimes**	
	CER	**WER**	**CER**	**WER**
CNN-1DLSTM (Moysset et al. [47])	11.52	35.64	6.14	20.11
MDLSTM (Pham et al. [7])	10.80	35.10	6.80	28.50
GNN-1DLSTM (Bluche et al. [9])*	10.17	32.88	5.75	19.74
2DLSTM (Moysset et al. [47])	8.88	29.15	4.94	16.03
2DLSTM-X2 (Moysset et al. [47])	8.86	29.31	4.80	16.42
CNN-Seq2Seq (Sueiras et al. [46])	8.80	23.80	4.80	15.90
CNN-Seq2Seq (Zhang et al. [21])	8.50	22.20	-	-
CNN-1DLSTM (Puigcerver et al. [13])	8.20	25.40	3.30	12.80
2DLSTM (Bluche et al. [48])	7.90	24.60	**2.90**	12.60
CNN-1DLSTM (Puigcerver et al. [13])*	7.73	25.22	4.39	14.05
CNN-Transformers (Kang et al. [49])	7.62	24.54	-	-
Deep BLSTM + Dropout (Bluche et al. [50])	7.30	24.70	5.60	20.90
2D-SACRN (Ours)	**6.76**	**20.89**	3.43	**11.92**

* Experiments run by Moysset et al. [47]

Japanese Text Recognition:

In the second experiment, we evaluated the performance of the 2D-SACRN model on the TUAT Kondate - offline handwritten Japanese text dataset in terms of CER and SER. To fairly compare with the previous models, we also do not use any data augmentation techniques as well as linguistic context information. Table 3.14 compares the recognition error rates by the 2D-SACRN model and the previous works of DCRN and AACRN

models on the test set of TUAT Kondate without using the language model. The 2D-SACRN model achieved CER of 2.49% and SER of 12.66% on the test set of TUAT Kondate. The results imply that the 2D-SACRN model obtains the state-of-the-art results on the TUAT Kondate dataset and outperforms the best model by about 10% of CER and 25% of SER.

Table 3.14. Recognition error rates (%) on the test set of TUAT Kondate.

Model	Kondate	
	CER	**SER**
Segmentation-based method [1]	11.2	48.53
DCRN_o-f&s	6.95	28.04
DCRN_o-s	6.44	25.89
End-to-End DCRN	3.65	17.24
AACRN	2.73	15.74
2D-SACRN (Ours)	**2.49**	**12.66**

Effects of 2D Self-Attention Mechanism:

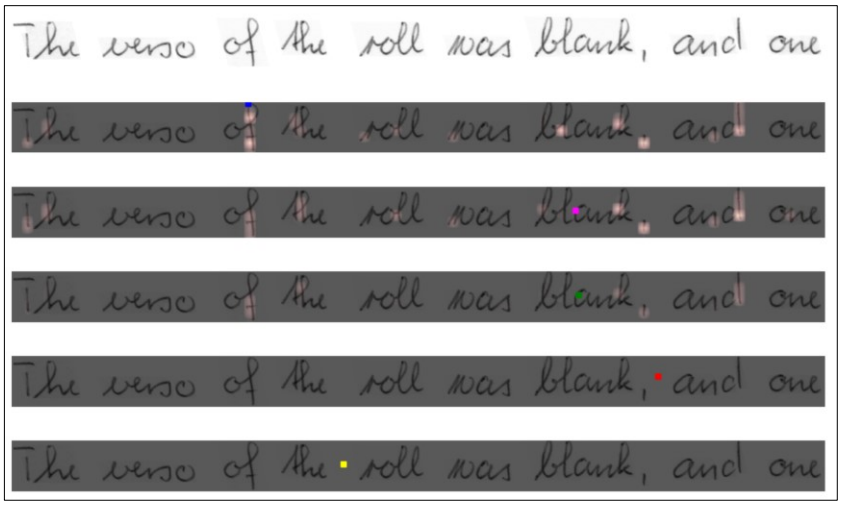
To measure the effectiveness of the 2D self-attention mechanism in the feature extractor of the 2D-SACRN, we prepared one variation, which was the same as the 2D-SACRN model except using the 2D self-attention block in the feature extractor. This variation is called 2D-SACRN_w/o_2DSelfAttn. We trained the 2D-SACRN_w/o_2DSelfAttn according to the same scheme applied to the 2D-SACRN model on the three datasets. Table 3.15 compares its recognition error rates with the 2D-SACRN model on the test set of the IAM Handwriting, Rimes, and TUAT Kondate datasets. In all datasets, the 2D-SACRN model slightly outperforms the 2D-SACRN_w/o_2DSelfAttn. The results imply that the 2D self-attention mechanism in the feature extractor improves the performance of the 2D-SACRN model for handwritten text recognition.

Table 3.15. Recognition error rates (%) with different feature extractors.

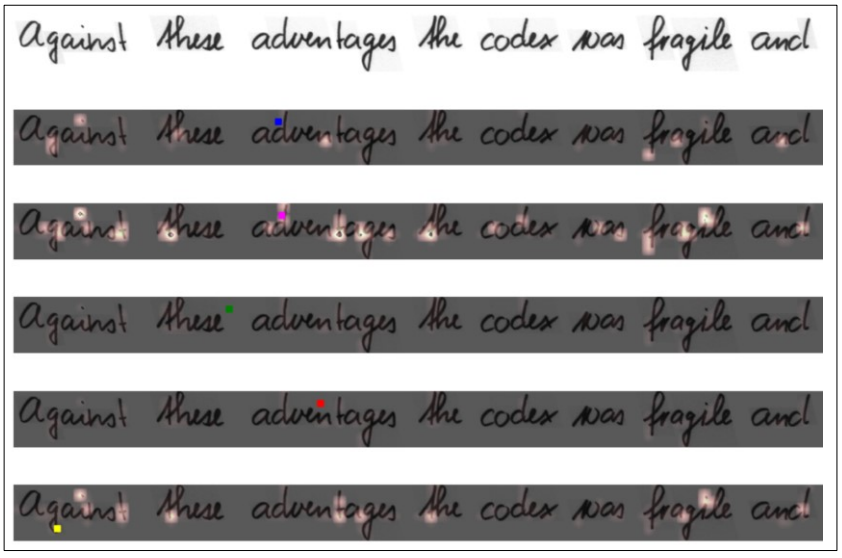
Model	IAM		Rimes		Kondate	
	CER	**WER**	**CER**	**WER**	**CER**	**SER**
2D-SACRN_w/o_2DSelfAttn	7.49	22.97	3.78	13.48	2.77	14.02
2D-SACRN	6.76	20.89	3.43	11.92	2.49	12.66

C. Visulization of 2D Self-Attention mechanism

To verify whether the 2D self-attention helps the feature extractor to capture the relationships between widely separated spatial regions in an input image, we visualize the 2D self-attention map in 2D-SACRN for different images in the IAM Handwritten dataset. Figure 3.14 shows the visualization of the 2D self-attention map for two images. In each group, the top image is the original input image, while each of the other five images shows one query point with color-coded dots (blue, fuchsia, green, red, and yellow) and the 2D



(a) A group of attention maps belonging to the first text image.



(b) A group of attention maps belonging to the second text image.

Figure 3.14. The visualization of 2D self-attention maps.

self-attention map for that query point. We observe that the 2D self-attention mechanism tends to focus on locations having similar texture to the query point, though these locations are far from the query point. For example, in the first group of Figure 3.14(a), the blue point (top of the “f” character) attends mostly to locations around the stroke of the “f”, “d”, “,”, and “k” characters. Besides, the fuchsia point (inside the stroke of the “a” character of the “blank”) do not attend to locations around its stroke but mostly attends to the stroke of the “f”, “d”, and “,” characters.

We also see that query points inside background regions seem not to attend mostly to any other location, such as the red and yellow points in Figure 3.14(a) as well as the green and red points in Figure 3.14(b). It seems because the points inside background regions do not mostly relate to any other location in the image. We also find that some query points are quite close in spatial location but have very different attention maps. For example, in the second group of Figure 3.14(b), the blue point and the fuchsia point are quite close but have very different attention maps. This shows that the adjacent points may freely attend to other distant locations. These observations demonstrate that the 2D self-attention mechanism helps the feature extractor to capture the relationships between widely separated spatial regions in an input image.

3.6. Text Line Image Generation Method

Deep Neural Networks, especially end-to-end models typically require a large data for training. However, for many handwriting datasets, especially handwritten Japanese character and text datasets, the number of data is not enough, so that it is necessary to apply data argumentation. Many data argumentation methods for handwriting datasets have been proposed by modifying the original data such as affine transformations [51, 52], nonlinear combinations [52, 53] and Random warp grid distortion [54]. However, such method just modifies the original data, can't gain the real text line image. In this work, we propose a synthetic pattern generation method which synthesize handwritten text line images from sentences in corpora and handwritten character patterns in the Nakayosi and Kuchibue [55] database with elastic distortions.

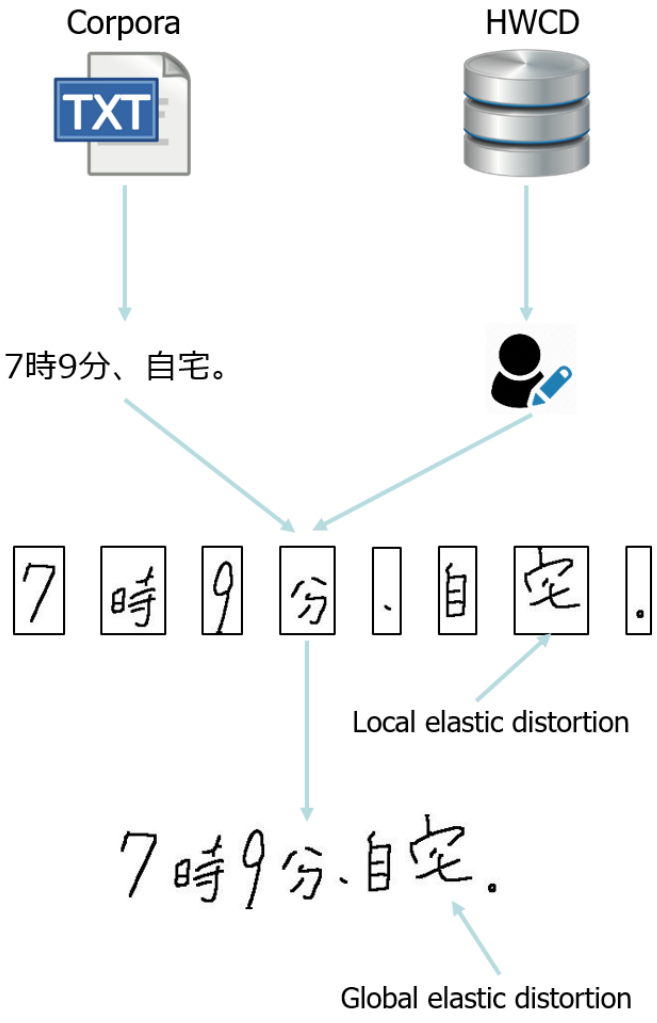


Figure 3.15. Synthetic pattern generation method.

3.6.1. Synthetic Data Generations.

Since Deep Neural Networks requires large data for training, we propose a synthetic pattern generation method which synthesizes handwritten text line images from sentences in corpora and handwritten character patterns in the isolated character database with local elastic distortion and global elastic distortion model. The overview of the synthetic pattern generation method is shown in Figure 3.15. The synthetic handwritten text line dataset is generated by taking the following 6 steps:

1. Get a sentence from the listed sentences of corpus.
2. Randomly choose a writer from the listed writers of the handwritten character pattern database.
3. For each character of the sentence in the step 1, a handwritten image of this character is randomly chosen from the writer selected in the step 2.
4. Apply a local elastic distortion to each handwritten pattern in the step 3.
5. Synthesize a handwritten text line image from the sentence selected in the step 1 and elastically distorted handwritten character images in the step 4 with random spacing between each character image.
6. Apply a global elastic distortion to the generated synthetic text line image.

3.6.2. Local Elastic Distortion.

The local elastic distortion model performs an affine transformation on each handwritten character image before concatenating them into a synthetic text line image. In the local elastic distortion model, we employ shearing, rotation, scaling, translation transformations.

Shear is a transformation that slants the shape of an object. There are two shear transformations include X-shear and Y-shear (vertical and horizontal shear). They are calculated by Eq. (3.6.1) and Eq. (3.6.2).

Translation is a transformation that moves an object to a different position without rotation. Scaling is a transformation that changes the size of an object. The translation and scaling transformations are shown in Eq. (3.6.3) and eq. (3.6.4).

Rotation is a transformation that rotates the object at particular angle α from its origin. The rotation transformation is shown in Eq. (3.6.5).

Here, (x', y') is the new coordinate of a point (x, y) transformed by any transformation model, α is the angle of the shear and rotation transformations, k is the scaling factor of the scaling transformation, the pair (t_x, t_y) is the shift vector of the translation transformation. The parameters of the local elastic distortion model is presented by $[(p_{SH}, \alpha), (p_T, t_x, t_y), (p_{SC}, k), (p_R, \alpha)]$, where p_{SH}, p_T, p_{SC} and p_R are the probabilities of applying the shearing, translation, scaling and rotation transformations, respectively, α is from -8^o *to* 8^o with a step of 0.1, t_x and t_y are from 3 to 5 pixels with a step of 1, and k is from 0.8 to 1.2 with a step of 0.01.

$$\begin{cases} x' = x + y\tan\alpha \\ y' = y \end{cases} \quad (3.6.1) \qquad \begin{cases} x' = x \\ y' = y + x\tan\alpha \end{cases} \quad (3.6.2)$$

$$\begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases} \quad (3.6.3) \qquad \begin{cases} x' = k\,x \\ y' = k\,y \end{cases} \quad (3.6.4)$$

$$\begin{cases} x' = x\cos\alpha - y\sin\alpha \\ y' = x\sin\alpha + y\cos\alpha \end{cases} \quad (3.6.5)$$

Figure 3.16 show examples of the local elastic distortion model with $\alpha = 8°$, k=0.9 and $t_x = t_y = 3$.

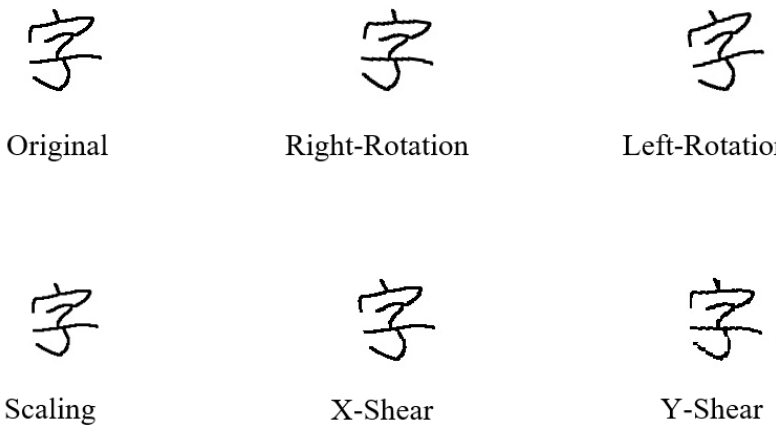


Figure 3.16. Examples of local elastic distortion by shearing, rotation and scaling transformations.

3.6.3. Global Elastic Distortion.

Global elastic distortion model performs affine transformation on a whole text line image generated by concatenating isolated handwritten character images. In the global elastic distortion, we employ the rotation and scaling transformations. The rotation and scaling transformations is similar to the local elastic distortion. The parameters of the global elastic distortion are presented by $[(p_{SC}, k), (p_R, \alpha)]$, where p_{SC} andp_R are the probabilities of applying the scaling and rotation transformations, respectively, k is the scaling factor and from 0.8 to 1.2 with a step of 0.01, and α is the angle of the global rotation transformation and from $-5^o \; to \; 5^o$ with a step of 0.1.

Figure 3.17 show examples of the global elastic distortion by the scaling and rotation transformations.

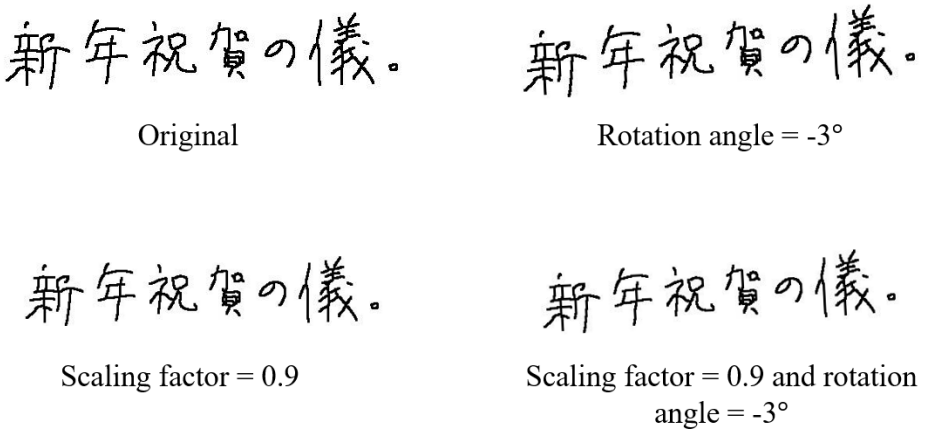


Figure 3.17. Examples of global elastic distortion by scaling and rotation

3.6.4. Experiments

To evaluate the effectiveness of the synthetic data generation method, we conducted experiments of the End-to-End DCRN model on standard benchmarks for offline handwritten Japanese text recognition. The information of the Synthetic Handwritten Text Line Dataset generated by the synthetic data generation method is given in Section A, the results of the experiments are presented in Section B, and the correctly recognized and misrecognized samples are shown in Sec. C.

A. Synthetic Handwritten Text Line Dataset.

Deep Neural Networks, especially end-to-end models typically require a large data for training. However, in handwritten Japanese text recognition, the current handwritten

Japanese text dataset TUAT Kondate consists of 13,856 text line images which just cover about 1,200 character categories (totally have 3345 character categories for JIS level-1). So, this dataset is quite small and the number of data is not enough to train the end-to-end DCRN model. So that we apply our proposed synthetic pattern generation method to argument the training data. We employ the sentences of Nikkei newspaper corpus and Asahi newspaper corpus and the handwritten character database, Nakayosi and Kuchibue [55] to generate the Synthetic Handwritten Text Line (SHTL) dataset. Nakayosi contains samples of 163 writers, 10,403 character patterns covering 4,438 classes per writer. Kuchibue contains handwritten samples of 120 writers, 11,951 character patterns covering 3,345 classes per writer. Nikkei corpus consists of about 1.1 million sentences collected from Nikkei News and Asahi corpus consists of about 1.14 million sentences collected Asahi News. We randomly choose 30,000 sentences which contain less than 30 characters from each corpus. Since it makes sure that the end-to-end model can be trainable by SHTL. SHTL consists about 60,000 of synthetic handwritten text line images. Figure 3.18 show samples of generated synthetic text line image in the SHTL dataset.

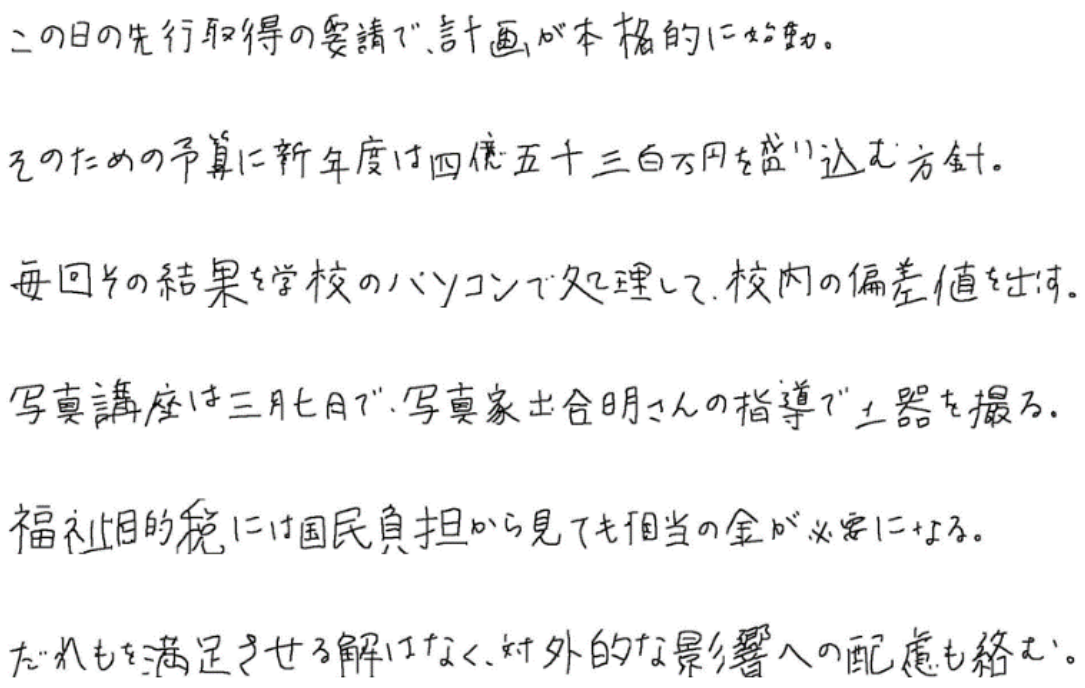


Figure 3.18. Samples of generated synthetic data.

B. Experiment Results

To evaluate the effectiveness of the synthetic data generation method, we train the End-to-End DCRN model by two datasets; the first is the training set of TUAT Kondate and

the second is the training set of TUAT Kondate combining the SHTL Dataset. We call the former End-to-End and the latter End-to-End_SHTL. We use the validation set and test set of TUAT Kondate to validate and test the performance of End-to-End and End-to-End_SHTL. Table 3.16 shows the recognition rate on the valid and test sets. End-to End_SHTL achieved LER of 1.95% and SER of 14.02% and outperformed the End-to-End on the test set of Kondate. These results show that the recognition accuracy is further improved when we use the SHTL dataset to train the end-to-end DCRN model. This means the synthetic data generation method improves the performance of the end-to-end DCRN model.

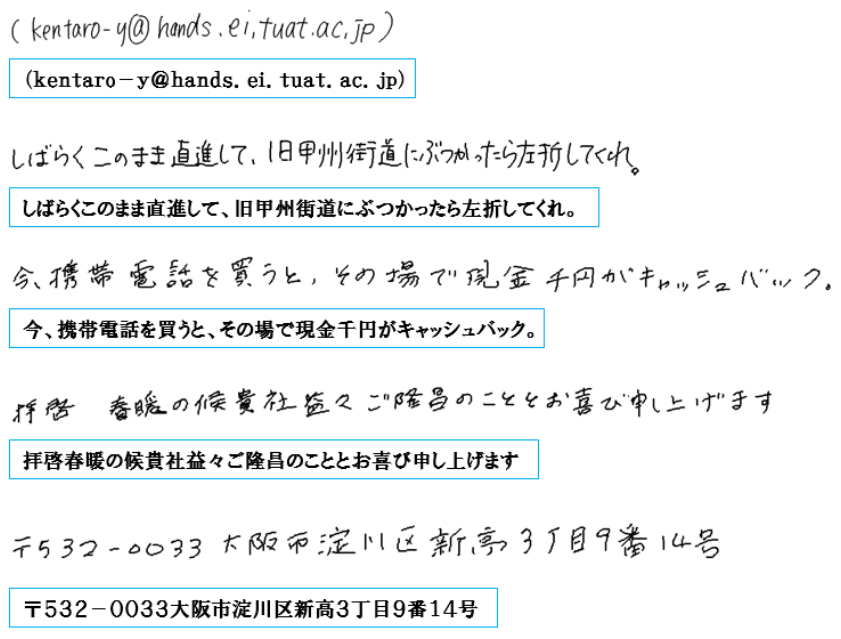
Table 3.16. Label Error Rate (LER) and Sequence Error Rate (SER) on Kondate.

Model	**LER**		**SER**	
	Valid set	**Test set**	**Valid set**	**Test set**
End-to-End	5.22%	3.65%	24.47%	17.24%
End-to-End_SHTL	3.62%	1.95%	21.87%	14.02%

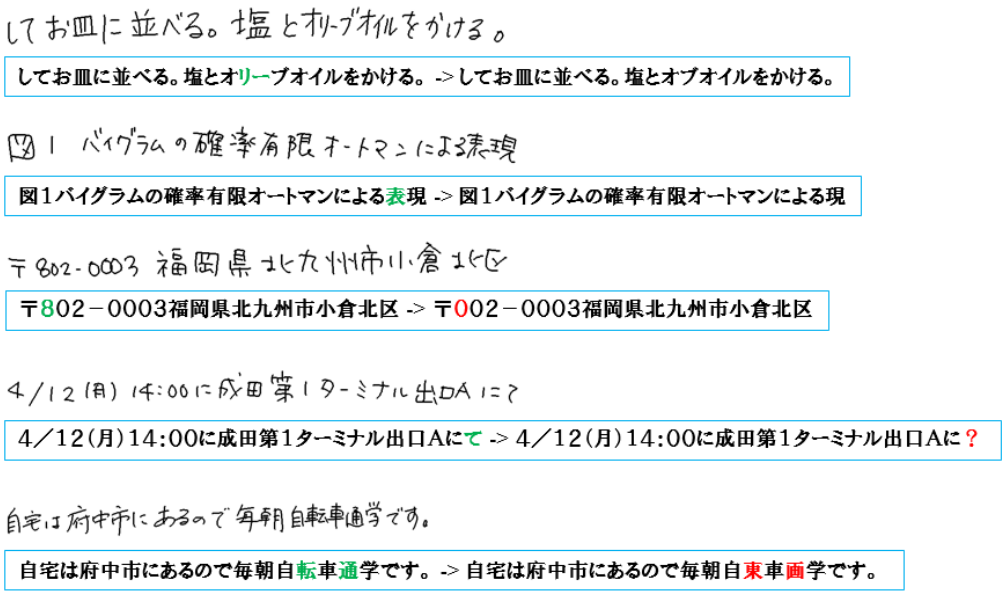
C. Correctly recognized and misrecognized samples

Figure 3.19 shows some correctly recognized and misrecognized samples by End-to-End_SHTL whose SER is about 14.02%. For each misrecognized sample, the upper image is an input handwritten text line image and the text bounded by the lower blue rectangular shows the ground-truth followed by "->" and the recognition resulted. There

are a total of 196 misrecognized samples among 1398 samples in the test set. Most of them are missing some characters in the ground-truth.



a). Correctly recognized samples.



b). Misrecognized samples.

Figure 3.19. Correctly recognized and misrecognized samples by End-to-End_SHTL.

3.7. Conclusions

In this chapter, we present models of Deep Convolutional Recurrent Network (DCRN) for recognizing offline handwritten Japanese text lines without explicit segmentation of characters. The DCRN model has three parts: a feature extractor by Convolutional Neural Network (CNN); an encoder by Bidirectional Long Short-Term Memory (LSTM); and a decoder by Connectionist Temporal Classification (CTC). As far as we know, this is the first approach that adopts DNNs for offline handwritten Japanese text recognition. The experiments show that the DCRN model outperforms the traditional segmentation-based method on the offline handwritten Japanese dataset – TUAT Kondate.

To solve the drawbacks of RNNs in the encoder, we propose an upgraded version of DCRN named Attention Augmented Convolutional Recurrent Network (AACRN) which introduces 1D self-attention mechanism in the encoder. The self-attention mechanism is complementary to RNN in the encoder and helps the encoder to capture long-range and multi-level dependencies across an input sequence. Experiments on the TUAT Kondate dataset show that the AACRN model has reduced the error rates drastically from the DCRN model. The experiments also show that the self-attention mechanism in the encoder improves the performance of the CRNN model for handwritten Japanese text recognition.

To solve the weakness of the CNN network in the feature extractor, we propose a 2D Self-Attention Convolutional Recurrent Network (2D-SACRN) model with a 2D self-attention mechanism for offline handwritten text recognition. In this model, we present a 2D self-attention mechanism in the feature extractor to help the CNN to capture the relationships between widely separated spatial regions in an input image. As far as we know, it is the first approach that employs the 2D self-attention mechanism in the feature extractor of the CTC-based model for offline handwritten text recognition. According to the extensive experiments on the three datasets of IAM Handwriting (English), Rimes (French), and TUAT Kondate, the 2D-SACRN model achieves similar or better accuracy than the state-of-the-art models. The 2D self-attention map visualization shows that the 2D self-attention mechanism helps the feature extractor capture the relationships between widely separated spatial regions in an input image.

Since the DCRN models require a large data for training, we propose a synthetic pattern generation method which synthesize handwritten text line images from sentences in corpora and handwritten character patterns in the isolated character database with elastic

distortions. The experiments on the offline handwritten Japanese text dataset – TUAT Kondate show that the synthetic pattern generation method improves the performance of the DCRN model.

Chapter 4. Japanese Historical Documents Recognition

4.1. Introduction

Under the support by the Center for Open Data in the Humanities (CODH) in Japan, the technical committee on Pattern Recognition and Media Understanding (PRMU) in the academic society of IEICE Japan held a contest to read deformed Kana in 2017 [20]. 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. The dataset for the contest consisting of three sub-datasets for the three levels is published[1]. We call the dataset Kana-PRMU. In this contest, we proposed the combination of a pre-trained CNN and an LSTM with CTC named by Deep Convolutional Recurrent Network (DCRN) for level 2 and the DCRN combined with a vertical line segmentation method for level 3 [21]. These methods won the best award with 12.88% character error rate (CER) for level 2 and 26.70% for level 3.

This chapter is based on our previous works which won the best algorithm award in the PRMU algorithm contest in 2017, but omits level 1 and focuses on level 2 and 3. Moreover, we added end-to-end-training and a two-dimensional Bidirectional Long Short-Term Memory (2DBLSTM) based model after the contest. We compare the pretrained 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 chapter is organized as follows: Section 4.2 presents methods for recognizing sequences of three vertically written Kana characters (level 2); Section 4.3 describes methods for recognizing unrestricted sets of Kana characters (level 3); Section

[1] http://www.iic.ecei.tohoku.ac.jp/~tomo/alcon2017/dataset.tar.gz

4.4 presents attention-based sequence to sequence methods for both level 2 and level 3 datasets; and Section 4.5 concludes the chapter.

4.2. Contest Overview

The PRMU contest is divided in three different levels (1 to 3) in accordance with the number of characters to be recognized [7]. All the tasks are to recognize Kana characters of 46 categories; Kanji characters are excluded. All characters are written with brushes. The Kana-PRMU dataset is compiled from 2,222 scanned pages of 15 pre-modern Japanese historical books and consists of three subsets for three levels. Figure 4.1 shows a sample page of the pre-modern Japanese books and examples of level 1, level 2, and level 3. The datasets for levels 1, 2, and 3 respectively consist of 228,334 segmented

Figure 4.1. Sample page in the contest [7].

single Kana images, 79,165 sequences of three vertically written Kana characters, and 12,583 samples of unrestricted Kana characters composed of three or more Kana characters, possibly in multiple lines. Character images are annotated with their bounding boxes and Unicodes. Contest participants can use only the provided datasets. The test sets to evaluate the submitted method are undisclosed.

As in other handwriting databases, there are large deformations and variations even in the patterns of the same category. Moreover, the old Kana uses several different notations for the same category, such as shown in Figure 4.2, where the categories ‘o’ and ‘ni’ have two and four notations, respectively. Furthermore, a notation of the category ‘u’ is similar to a notation of the category ‘ka’ as shown in Figure 4.3. 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 Figure 4.4; smeared and stained as shown in Figure 4.5. Their backgrounds are often neither uniform nor even as shown in Figure 4.6.

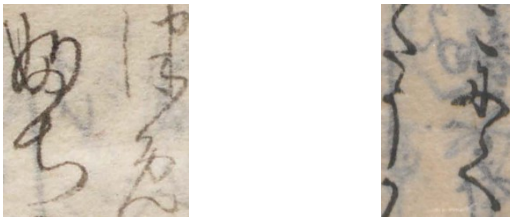
(a) Two notations of category ‘o’

(b) Four notations of category ‘ni’

Figure 4.2. Different notations of the same category.

(a) Notation of category ‘u’ (b) Notation of category ‘ka’

Figure 4.3. Similar notations between different categories.



(a) Fade and show-through. (b) With width variations of vertical lines.

Figure 4.4. Fade and show-through.

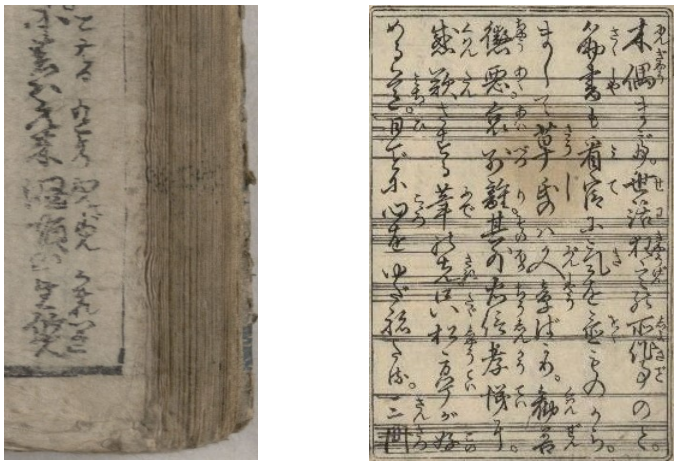


Figure 4.5. Fragmented patterns and noisy patterns.

Figure 4.6. Various backgrounds.

In this work, we focus on the level 2 and 3 subsets. Since PRMU did not publicize the test set of the contest, we use one of the 15 historical books as the test set. The remaining 14 books are used for training and validation. Among the 15 books, the 15th book contains many noisy patterns and a variety of backgrounds. Thus, the 15th book is selected as the test set for levels 2 and 3. The other books are divided randomly to form the training and validation sets with the ratio of 9:1. Note that the text is written vertically.

4.3. Three Kana Sequence Recognition

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

4.3.1. Level 2 Dataset

The level 2 dataset consists of 79,165 images of single vertical text line composed of three Kana. All images in the 15th book are used as the test set. The other images are divided randomly from the training and validation sets with a 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. Since each image has only a single vertical text line, all images are linearly scaled to the same width (96 pixels) with arbitrary lengths before fed to the recognition system. Figure 4.7 shows some vertical text line images in the dataset.

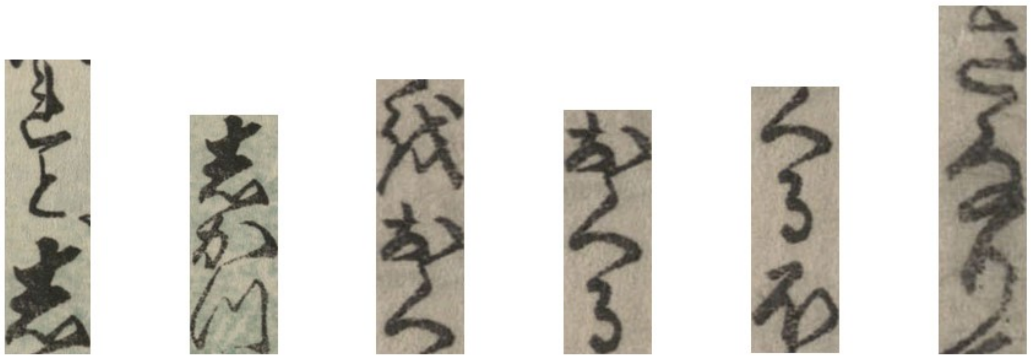


Figure 4.7. Some vertical text line images in the level 2 dataset.

4.3.2. Methods for level 2

In level 2, we employ the DCRN model which is mentioned in the chapter 4 for recognizing three-Kana-character sequence images. The DCRN model consists of three main parts: a feature extractor by s CNN; an encoder by BLSTM; and a decoder by a CTC as shown in Figure 4.8.

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 encoder 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.

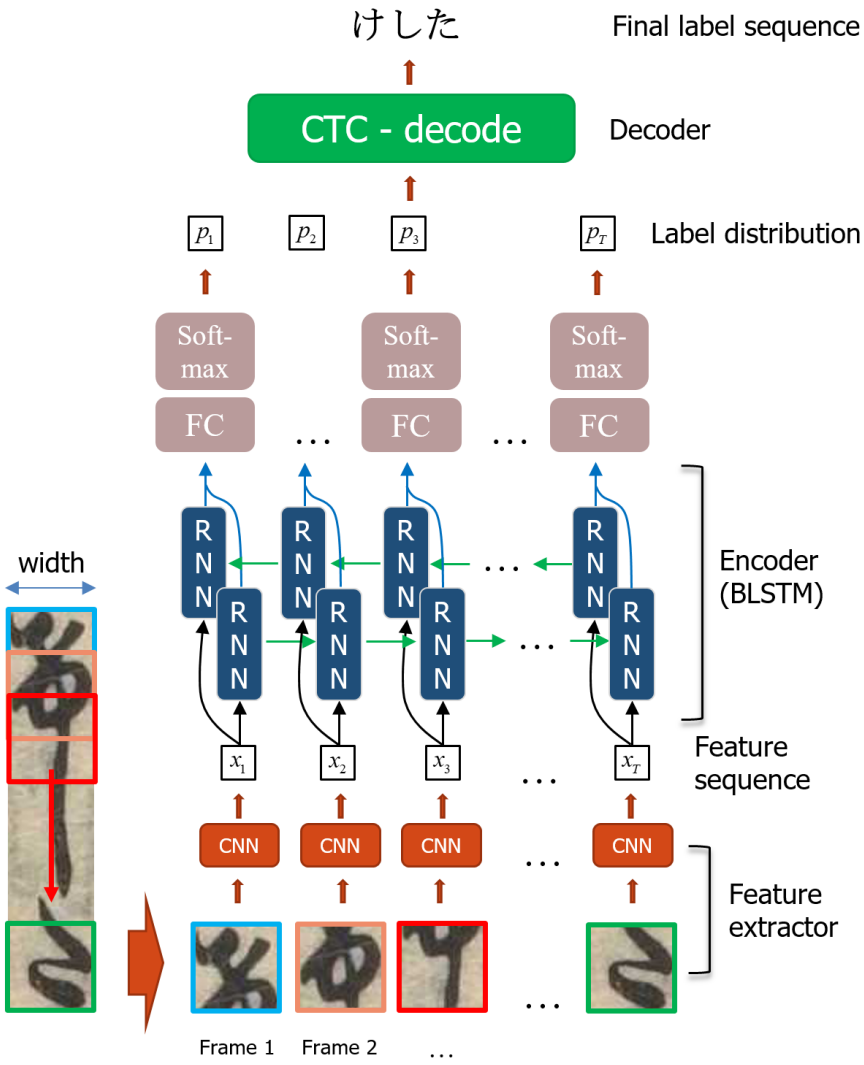


Figure 4.8. Network architecture of DCRN.

4.3.3. Implementation Details

For the contest, we employed 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. The following sections A and B 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.

A. 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 4.1 in which 'maps,' 'k,' 's,' and 'p' denote the number of kernels, kernel size, stride, and padding size of convolutional layers, respectively.

Table 4.1. Network configuration of our CNN model.

Type	Configurations
Input	64×64 image
Conv1 - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling1	#window:2×2, s:2×2
Conv2 - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling2	#window:2×2, s:2×2
Conv3 - ReLu	#maps:128, k:3×3, s:1, p:1
MaxPooling3	#window:2×2, s:2×2
Conv4 - ReLu	#maps:128, k:3×3, s:1, p:1
MaxPooling4	#window:2×2, s:2×2
Conv5 - ReLu	#maps:256, k:3×3, s:1, p:1
MaxPooling5	#window:2×2, s:2×2
FC1 - ReLu	#nodes: 200
FC2 - ReLu	#nodes: 200
Softmax	#nodes: 46(number class)

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 64×64 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 4.9 shows this way of forming the feature extractor.

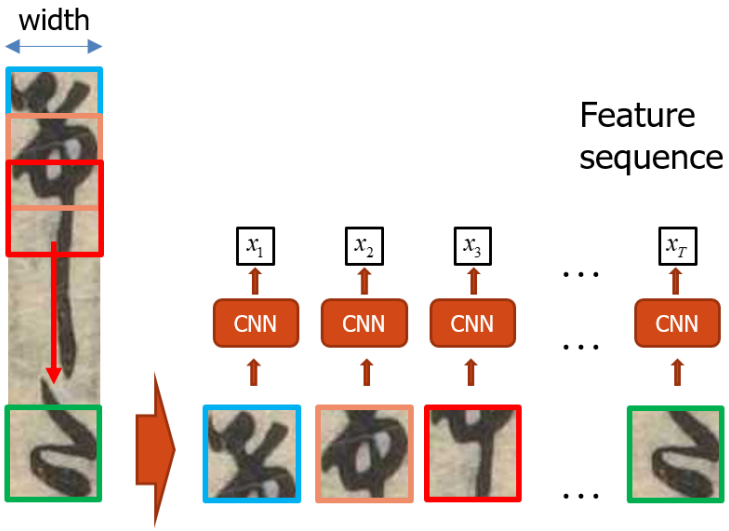


Figure 4.9. Convolutional feature extractor in DCRN-o.

The second method employs a sub-window of 64×32 pixels and the sliding stride size of 32 pixels (without overlap sliding) as shown in Figure 4.10 and applies the CNN network without the softmax and full connection layers to 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.

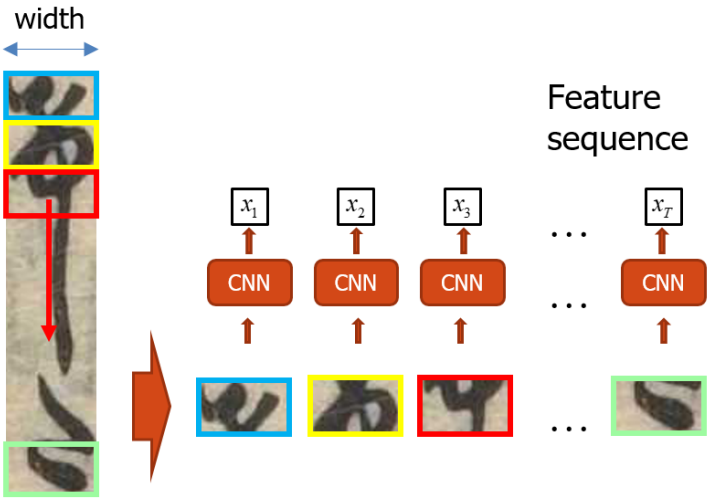


Figure 4.10. Convolutional feature extractor in DCRN-wo.

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 4.11 shows its architecture, where 'h' and 'w' denote the height and width of an input image and 'h' ' and 'w'' denote the height and width of an output image. We call this method DCRN-ws.

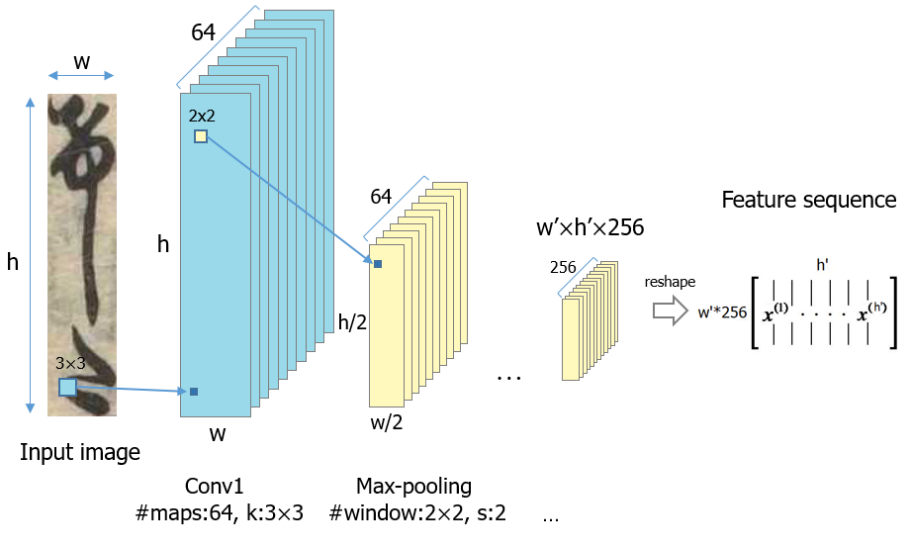


Figure 4.11 Convolutional feature extractor in DCRN-ws.

For the encoder, we employ three levels of 1DBLSTM 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 encoder. Here the number of category is added one category for blank character. The classifier is trained using the online steepest decent 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.

B. 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 4.2 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.

Table 4.2. Network configuration of our CNN model.

Type	**Configurations**
Input	h×w image
Conv1 - Batch Norm - LReLu	#maps:32, k:3×3, s:1, p:1
MaxPooling1	#window:2×2, s:2×2
Conv2 - Batch Norm - ReLu	#maps:32, k:3×3, s:1, p:1
MaxPooling2	#window:2×2, s:2×2
Conv3 - Batch Norm - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling3	#window:2×2, s:2×2
Conv4 - Batch Norm - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling4	#window:1×2, s:1×2

At the encoder, 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.

4.3.4. 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 in Eq. (3.3.2) and Eq. (3.3.3).

Table 4.3 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 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 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 4.3 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.

Table 4.3. Recognition error rates (%) on level 2 dataset.

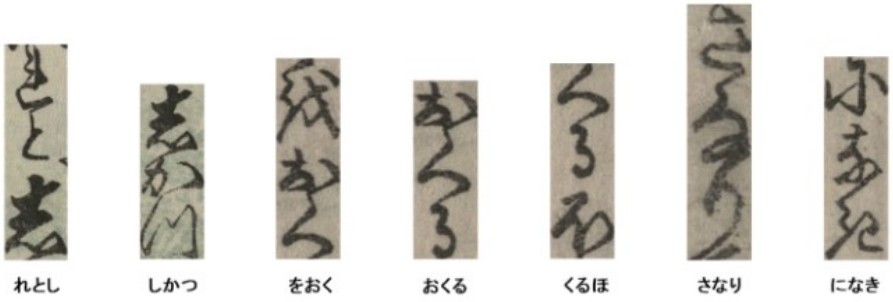
Networks	**LER**		**SER**	
	Valid set	**Test set**	**Valid set**	**Test set**
DCRN-wo	14.19	26.79	33.51	59.28
DCRN-ws	10.21	18.56	25.07	44.81
DCRN-o_16	9.72	14.44	23.62	35.11
DCRN-o_12	8.65	12.88	21.03	31.60
End-to-End DCRN_ws	5.10	**10.90**	13.10	**27.70**

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.

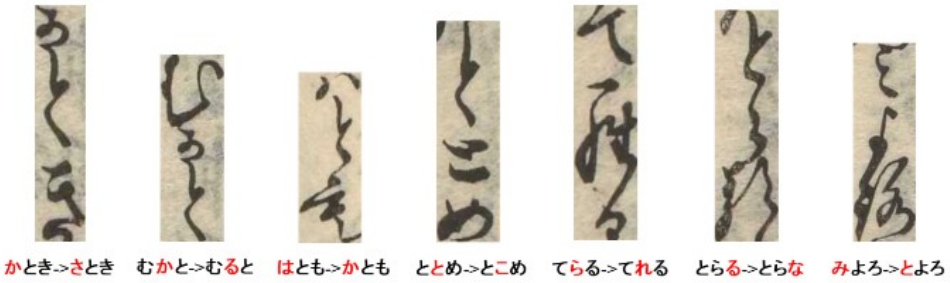
On the other hand, there are still large gaps between the validation and testing sets. This suggests that the number of training samples was not adequate, so overfitting occurred.

Employing more samples for training or applying data augmentation may decrease the error rates to some extent.

Figure 4.12 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. There are 5,320 misrecognized samples among 16,835 samples. Most are misrecognized due to only one of the three characters.



(a) Correctly recognized samples



(b) Misrecognized samples

Figure 4.12. Samples recognized and misrecognized by DCRN-o_12.

4.3.5. 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:

$$\text{Avg}(error\_rate) = \sum_{i=1}^{5} \frac{error\_rate_i \times N_i}{N} \quad (4.3.1)$$

where $error\_rate_i$ 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 4.4 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 4.3. 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 4.4. 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.

Table 4.4. Recognition error rates (%) of five models.

Models	**Test set**		
	LER	**SER**	**Number of samples**
Model 1	16.05	38.41	25,493
Model 2	8.82	23.31	7,976
Model 3	23.87	52.60	9,025
Model 4	14.36	30.82	2,609
Model 5	12.09	30.44	34,062
Average	14.45	34.44	-

4.4. Unrestricted Kana Recognition

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

4.4.1. 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. 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 above, there are some other difficulties such as the vertical and horizontal guide lines (Figure 4.13), the overlap or even touching between two vertical lines (Figure 4.14), and fade and show-through (Figure 4.15). In Figure 4.14(a), we draw blue bounding boxes to show each character. In the experiments, we use each image in its original size.

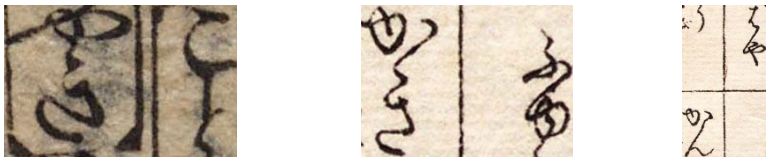
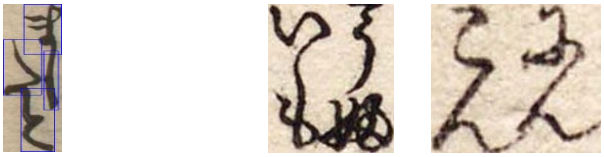


Figure 4.13. Level 3 images containing vertical and horizontal guide lines.



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

Figure 4.14. Overlap or touching between two vertical lines.

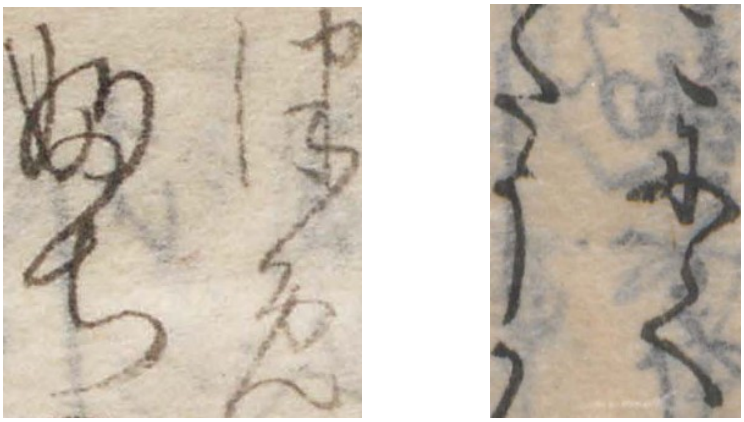


Figure 4.15. Fade and show-through.

4.4.2. 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.

A. 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. Figure 4.16 shows the process of methodology for recognizing unrestricted kana in level 3.

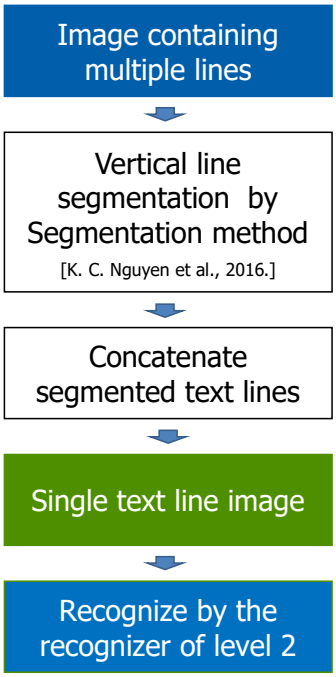


Figure 4.16. The methodology for recognizing unrestricted kana in level 3.

1) Vertical text line segmentation and concatenation

For vertical text line segmentation, we employ the segmentation method [13] tuned to vertical writing. Since there are many noises after binarization in historical documents, we remove connected components (CCs) that have areas smaller than the threshold of 25 pixels (5×5). The size of the *i*-th connected component (*Si*) is calculated from the average of the height and the width of its bounding box. Each 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.

$$AS = \frac{2}{N}\sum_{i=\frac{N}{2}}^{N} S_i \qquad (4.4.1)$$

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 4.17 shows some generated text line images.

Figure 4.17. Concatenated text lines.

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.

B. 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.

C. 2DBLSTM approach

The third approach replaces the CNN in the second approach by a stage of 2DBLSTM with the result of three 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 2×2, 4×2, and 4×2 for this sequence of stages, respectively. The input window sizes are applied to reduce the dimensionality of input images. Two feedforward full-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.

4.4.3. Experiments on level 3

Table 4.5 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 4.5 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 × 3 characters contributes to learning these models.

Table 4.5. Recognition error rates (%) on level 3 dataset.

Networks	**LER**		**SER**	
	Valid set	**Test set**	**Valid set**	**Test set**
CNN + 2DBLSTM_b:64_b:128_Lv3	14.45	44.18	55.82	97.16
CNN + 2DBLSTM_b:64_b:128_Lv2&3	23.59	43.09	67.38	94.55
2DBLSTM_b:2_b:10_b:50_Lv3	18.59	46.73	66.37	98.81
2DBLSTM_b:2_b:10_b:50_Lv2&3	15.55	37.72	63.35	94.55
Seg + DCRN-o_12_Lv3	11.72	26.70	49.14	82.57
Seg + DCRN-o_12_Lv2&3	9.47	24.24	40.53	78.81
Seg + End-to-End DCRN_ws_Lv3	4.30	18.50	21.50	73.70
Seg + End-to-End DCRN_ws_Lv2&3	2.80	**12.30**	15.40	**54.90**

Figure 4.18 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.

a) Correctly recognized samples.

b) Misrecognized samples.

Figure 4.18. Samples recognized and misrecognized by DCRN-o_12_Lv3.

4.4.4. Cross validation of Seg plus End-to-End DCRN_ws

In the same way as in Section 4.3, 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 4.6 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 4.5. This result is the same

as that in Section 4.3.5. Large variation in the performance is again the same as that in Section 4.3.5, possibly for the same reason.

Table 4.6. Recognition error rates (%) of five models.

Models	**Test set**		
	LER	**SER**	**Number of samples**
Model 1	24.19	75.73	3,433
Model 2	16.66	66.44	1,353
Model 3	35.84	87.25	2,709
Model 4	19.76	51.14	617
Model 5	18.72	75.57	4,471
Average	23.73	75.95	-

4.5. Conclusion

In this chapter, we 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 12.30% 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.

Chapter 5. Attention-based Model for Multiple Text Line Recognition

5.1. Introduction

This chapter introduces an attention-based row-column encoder-decoder (ARCED) model for recognizing multiple text lines image in Japanese historical documents. Since Japanese historical documents were written cursively through an entire text line with neighbor text lines touching each other, a segmentation-free approach is sought. We propose a model consisting of three main parts: a feature extractor, a row-column encoder, and a decoder. Given an input image, the feature extractor extracts a feature grid from it by a CNN. The row-column encoder applies a row bidirectional LSTM (BLSTM) and a column BLSTM to encode the feature grid in the horizontal direction and the vertical direction, respectively. The decoder applies an attention-based LSTM to generate the final target text based on the attended pertinent features. In this model, we incorporate a row-column BLSTM in the encoder to capture the sequential order information in both the vertical and the horizontal directions and a residual LSTM network in the decoder to take advantage of entire past attention information.

Experiments on level 2 and level 3 of the Kana-PRMU dataset show that the ARCED model reduces the error rates for single text lines (level 2) and for three or more characters possibly in multiple lines (level 3) drastically from the previous methods [7]. The experiments also show that the row-column BLSTM in the encoder and the residual

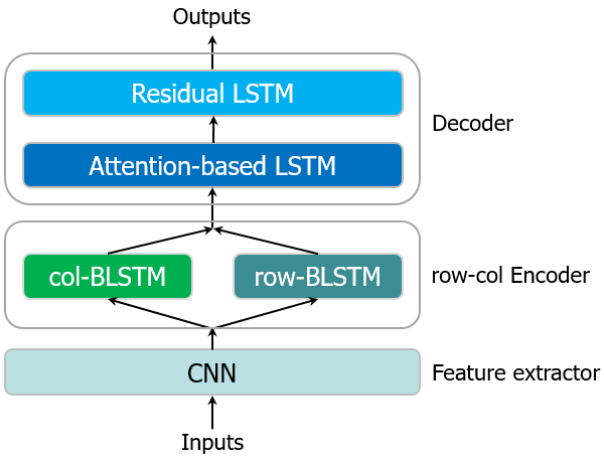


Figure 5.1. The overview of the ARCED model.

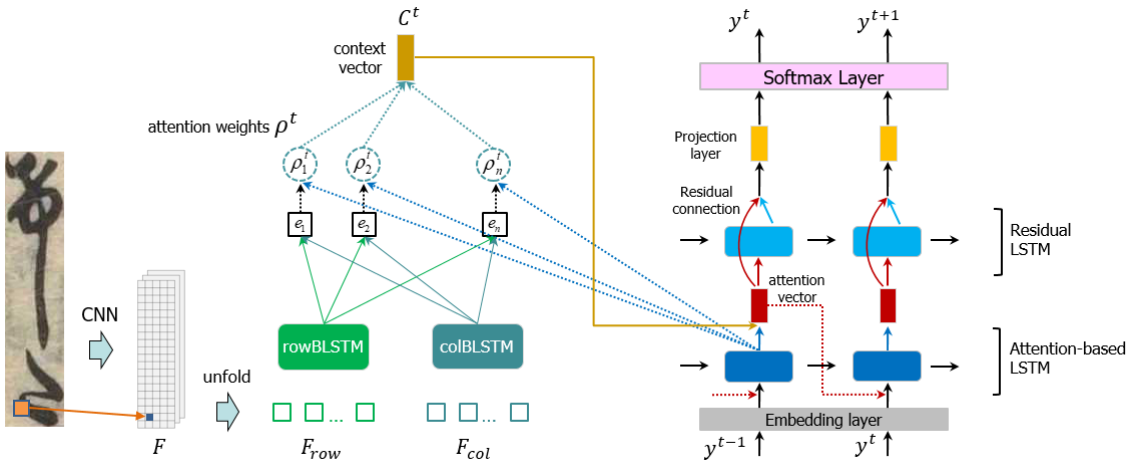


Figure 5.2. Network architecture of the ARCED model.

LSTM network in the decoder improve the performance of the attention-based encoder-decoder model for the text recognition task in the Japanese historical documents.

The contributions of this model are three folds. First, we present an attention-based encoder-decoder model for recognizing multiple text lines in Japanese historical documents. Second, we propose a row-column BLSTM in the encoder to encode the grid feature in both the vertical and horizontal directions. Third, we introduce a residual LSTM network in the decoder to take advantage of the entire past attention information.

The rest of this chapter is organized as follows. Section 5.2 presents an overview of the ARCED model. The experiments are reported in Section 5.3. Finally, conclusions are presented in Section 5.4.

5.2. The Proposed Method

We propose an attention-based row-column encoder-decoder (ARCED) model consisting of three main parts: a feature extractor, a row-column encoder and a decoder as shown in Figure 5.1. From the bottom of the ARCED, the feature extractor extracts a feature grid from the input image by DCNN. Then, the row-column encoder applies a row BLSTM and a column BLSTM to encode the feature grid in the horizontal and vertical directions, respectively. At the top of ARCED, the decoder applies an attention-based LSTM and a residual LSTM to focus on the pertinent encoded features and generates the final target text. Figure 5.2 shows the detail of ARCED. The parameters ρ^t,C^t, and y^t denote the attention weights, the context vector and the output at time *t* of the decoder, respectively. We describe the details of each part in the following sections.

A. Feature Extractor

CNN is a class of Deep Neural Network designed to work well with the 2D structure of an input image. A standard architecture of CNN consists of a number of convolutional and pooling layers optionally followed by fully connected layers and a softmax layer. CNNs have also been demonstrated to be compelling network architectures for feature extraction [38, 39].

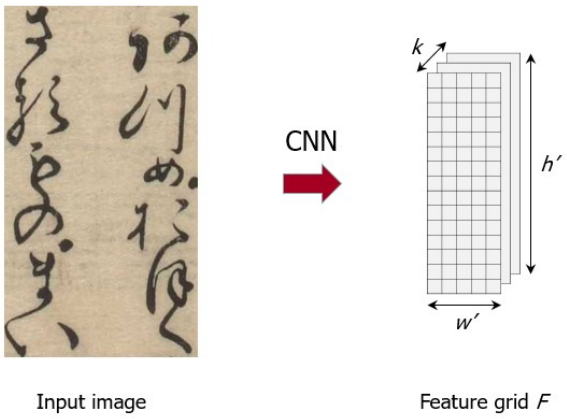


Figure 5.3. Feature extraction for a gray-scale input image (c=1).

In this work, we employ a standard CNN to build the feature extractor component. The CNN is constructed by taking convolutional, max-pooling layers while fully connected and softmax layers are removed. As shown in Figure 5.3, we apply the feature extractor to extract visual features from an input image of the size $\{h, w, c\}$, where h' and w' are the height and width of the input image and c is the color channel, resulting in a feature grid F of the size $\{h', w', k\}$, where h' and w' are the height and width of the feature map and k denotes the number of channels. The feature grid F is fed to the encoder.

B. Row-column Encoder

LSTM introduced by S. Hochreiter et al. [40] is a special kind of Recurrent Neural Network (RNN) designed to address the vanishing gradient problem when learning input sequences with long-range dependencies. Several variants of the LSTM architecture have been proposed, such as the Gated Recurrent Unit (GRU) in [56]. A. Graves et al. [57] described a variant of LSTM, which is most commonly used in literature and demonstrated that it outperforms the other variants for the IAM dataset of online handwritten text lines [58]. In this work, we employ the variant of LSTM in [57] to build the encoder and decoder components.

The previous work for scene text recognition [27] and math recognition [59] use a row encoder which encodes the feature gird F by running one BLSTM network across each row in the grid. Therefore, it cannot capture the sequential order information in the vertical direction. In order to capture the sequential order information in both the vertical and horizontal directions, we propose a row-column encoder, which consists of two BLSTM networks. The first one runs across each row in the feature grid to capture the sequential order information in the horizontal direction, while the second one runs across each column in the grid to capture the sequential order information in the vertical direction as shown in Figure 5.4. We refer to the row BLSTM as $\mathrm{BLSTM}_{\mathrm{row}}$ and the column BLSTM as $\mathrm{BLSTM}_{\mathrm{col}}$. As shown in Figure 5.4, $\mathrm{BLSTM}_{\mathrm{row}}$ and $\mathrm{BLSTM}_{\mathrm{col}}$ encode the feature grid F to get the encoded feature grid E_{row} and E_{col}, respectively. Then, E_{row} and E_{col} are concatenated to form $E_{\mathrm{row\&col}}$, which is unfolded from the top to the bottom and from the right to the left to produce a sequence of encoded vectors $E = (e_1, e_2 \cdots e_n)$, where n is the number of feature vectors. Then, we have:

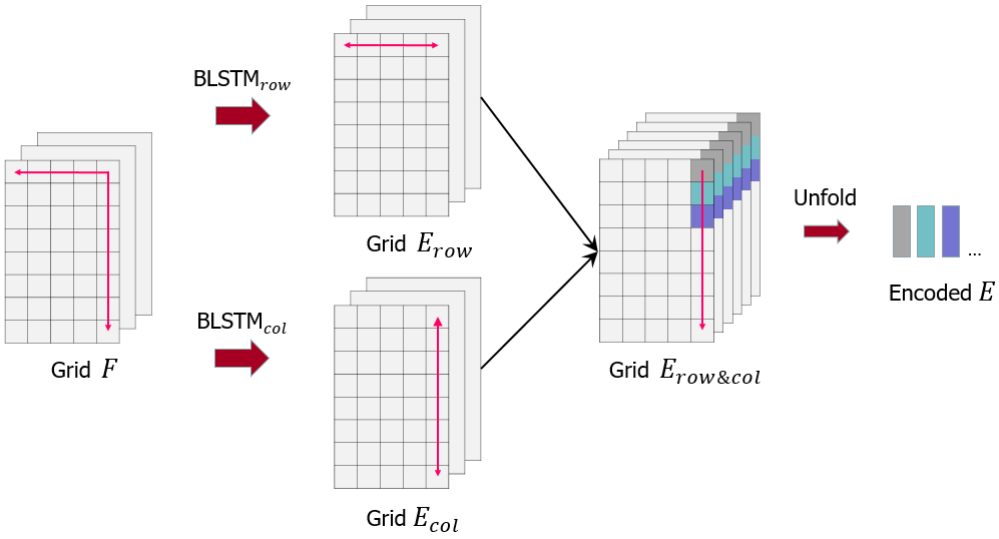


Figure 5.4. Row-column BLSTM encoder.

C. Attention-based decoder

In the original attention-based seq2seq model [24], the decoder consists of two parts: an attention mechanism and an attention-based LSTM network. At each time step t in the decoding phase, the attention weights ρ^t are calculated from the encoded vectors and the target hidden state h^t. Given the attention weights, the context vector c^t is computed as the weighted average over all encoded features. Then the attention vector a^t is computed

by concatenating the context vector c^t with the target hidden state h^t. Finally, the attention vector a^t is fed through the softmax layer to produce a predictive distribution y^t as shown in Eq. (5.2.1):

$$y^t = \text{softmax}\left(W_a \times a^t\right) \tag{5.2.1}$$

In the original attention-based models, after deciding which location to pay attention to, predictive distributions are made based only on the current attention vector. However, to be effective, all past attention vectors (predictive distributions) should be maintained during the decoding process to keep track of which characters have been predicted. To remedy this problem, we introduce an LSTM network between the attention vector and the softmax layer, as shown in Figure 5.1. This LSTM network compresses all past attention vectors and the current attention vector into a fixed-length vector. Then, it is fed through the softmax to produce predictive distributions. Consequently, the system takes advantage of all past attention vectors when producing the current predictive distributions.

Motivated by the idea of the residual connection in [41, 60], we add a residual connection between the attention vector and the softmax layer, as shown in Figure 5.2. Then, we denote the LSTM between the attention vector and the softmax layer by residual LSTM. The residual connection can help the model to address the exploding and vanishing gradient problem. Let LSTM_{Res} be the residual LSTM network which has the residual connection. At the *t*-th time step, we have:

$$h^t_{\text{LSTM}}, O^t_{\text{LSTM}} = \text{LSTM}\left(h^{t-1}_{\text{LSTM}}, a^t\right) \tag{5.2.2}$$

$$O^t_{\text{Res}} = O^t_{\text{LSTM}} + a^t \tag{5.2.3}$$

where $\boldsymbol{a^t}$ is the attention vector at the time step *t*, $\boldsymbol{h^t_{\text{LSTM}}}$ and $\boldsymbol{O^t_{\text{LSTM}}}$ are the hidden states and the output of the LSTM at the time step *t*, respectively; and $\boldsymbol{O^t_{\text{Res}}}$ is the output of the residual LSTM network.

We now describe how our decoder works. At each time step *t* in the decoding phase, the attention-based decoder generates one character y^t based on the current output O^t of the residual LSTM network as shown in Eq. (5.2.4):

$$y^t = \text{softmax}\left(W_o \times O^t\right) \tag{5.2.4}$$

The current output O^t is computed by the previous hidden state of the residual LSTM network h_{Res}^{t-1} and the current attention vector a^t as shown in Eq. (5.2.5):

$$h_{\text{Res}}^t, O^t = \text{LSTM}_{\text{Res}}\left(h_{\text{Res}}^{t-1}, a^t\right) \tag{5.2.5}$$

The current attention vector a^t is calculated from the concatenation of the current hidden state h_{Attn}^t and the current context vector c^t as shown in Eq. (5.2.6):

$$a^t = \tanh\left(W_c\left[c^t; h_{\text{Attn}}^t\right]\right) \tag{5.2.6}$$

The hidden state h_{Attn}^t is computed as shown in Eq. (5.2.7):

$$h_{\text{Attn}}^t = \text{LSTM}_{\text{Attn}}\left(h_{\text{Attn}}^{t-1}, \left[\text{Embed}(y^{t-1}), a^{t-1}\right]\right) \tag{5.2.7}$$

where $\text{LSTM}_{\text{Attn}}$ is the attention-based LSTM network, y^{t-1} is the previous ground-truth symbol (when training the model) or the previously predicted symbol (when testing the model), Embed is an embedding layer and a^{t-1} is the previous attention vector.

The current context vector c^t is computed by the current attention weights (attention probabilities) $\rho^t = (\rho_1^t, \rho_2^t \ldots \rho_n^t)$ and the sequence of encoded vectors $E = (e_1, e_2 \cdots e_n)$ as shown in Eq. (5.2.8):

$$c^t = \sum_{i=1}^{n} \rho_i^t e_i \tag{5.2.8}$$

The current attention weights ρ^t is calculated based on the hidden state h_{Attn}^t of LSTM_{Attn} and the sequence of encoded vectors E as shown in Eq. (5.2.9) and (5.2.10):

$$\rho_i^t = \frac{\exp\left(\text{score}\left(h_{\text{Attn}}^t, e_i\right)\right)}{\sum_{j=1}^{n} \exp\left(\text{score}\left(h_{\text{Attn}}^t, e_j\right)\right)} \tag{5.2.9}$$

$$\text{score}\left(h_{\text{Attn}}^t, e_i\right) = \tanh\left(w_h h_{\text{Attn}}^t + w_e e_i\right) \tag{5.2.10}$$

The decoding process is repeated until the decoder produces an <END> (end token). The ARCED model can be end-to-end trained by the gradient descent algorithm with a standard cross-entropy loss function.

5.3. Experiments

To verify the effectiveness of each part of the ARCED model and compare its performance with other methods, we conducted experiments on the level 2 and level 3 subsets of Kana_PRMU. The implementation details are described in Section 5.3.1, the results of the experiments are presented in Section 5.3.2, and the analysis on recognized and misrecognized samples is given in Section 5.3.3.

5.3.1. Implementation Details

The architecture of the CNN model used to build the feature extractor is shown in Table 5.1, where '#maps', 'k', 's' and 'p' denote the number of kernels, the kernel size, the stride size and the padding size of each convolutional layer, respectively. It consists of 6 convolutional layers. Batch normalization is applied to all convolutional layers. Each convolutional layer in the first five convolutional layers is followed by Max-Pooling layers.

Table 5.1. Network configuration of our CNN model.

Type	Configurations
Input	h×w image
Conv1 - Batch Norm - ReLu	#maps:32, k:3×3, s:1, p:1
MaxPooling1	#window:2×2, s:2×2
Conv2 - Batch Norm - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling2	#window:2×2, s:2×2
Conv3 - Batch Norm - ReLu	#maps:64, k:3×3, s:1, p:1
MaxPooling3	#window:2×2, s:2×2
Conv4 - Batch Norm - ReLu	#maps:128, k:3×3, s:1, p:1
MaxPooling4	#window:1×2, s:1×2
Conv5 - Batch Norm - ReLu	#maps:256, k:3×3, s:1, p:1
MaxPooling5	#window:2×1, s:2×1
Conv6 - Batch Norm - ReLu	#maps:256, k:3×3, s:1, p:1

At the row-column encoder, both the row BLSTM network $BLSTM_{row}$ and the column BLSTM network $BLSTM_{col}$ are composed of forward and backward layers where each forward or backward layer is a single LSTM layer having 256 hidden nodes. To prevent overfitting when training, we apply the dropout (drop rate = 0.2) in all LSTM layers of the BLSTM networks.

At the decoder, the attention-based LSTM network LSTM_Attn consists of two LSTM layers of 512 hidden nodes and the residual LSTM network LSTM_Res consists of a single LSTM layer of 512 hidden nodes. A projection layer and a softmax layer with the node size equal to the character set size plus the start token and the end token are applied after the residual LSTM network. The entire ARCED model is end-to-end trained using Adam [35] with the learning rate of 0.001 and the batch size of 4. We do not use any data augmentation or data preprocessing technique in the training process. The training process stops when the recognition accuracy of the validation set does not gain after ten epochs.

5.3.2. Experiment Results

In order to evaluate the performance of the ARCED model, we employ the terms of Character Error Rate (CER) and Sequence Error Rate (SER) that are defined in Eq. (3.3.1) and Eq. (3.3.2).

A. Single line recognition

First, we conducted an experiment on the level 2 subset, which consists of single vertical line images of three Kana characters. Table 5.2 compares the recognition error rates by the ARCED model and the state-of-the-art DCRN models reported in Section 5.3 on the test set of the level 2 subset.

The ARCED model achieved CER of 4.15% and SER of 11.43% on the test set, which reduced CER to one third and SER to half compared with the best DCRN method. The results imply that ARCED outperforms the state-of-the-art recognition accuracy of the DCRN models.

We offer some possible explanations for this improved performance. First, the attention mechanism seems to help the model focus on the pertinent features to predict characters. Second, the decoder helps the model learn the context statistics in the training vocabulary since the decoding mechanism is similar to the language model (given a sequence of previous characters, it predicts the next character). Third, the row-column BLSTM in the encoder and the residual LSTM in the decoder also improve the performance of the system. We will elaborate on these reasons in Sections C and D below.

Table 5.2. Recognition error rates (%) on level 2 test set.

Model	CER	SER

DCRN-wo	26.79	59.28
DCRN-ws	18.56	44.81
DCRN-o_16	14.44	35.11
DCRN-o_12	12.88	31.60
End-to-End DCRN	10.90	27.70
ARCED	**4.15**	**11.43**

Figure 5.5 shows correctly recognized samples and their visualization of attention weights. We can see that the attention mechanism focuses from the top of the image to read the first character, and shifts one character down to read the next character and so on. This mechanism is similar to how humans read a vertical text line.

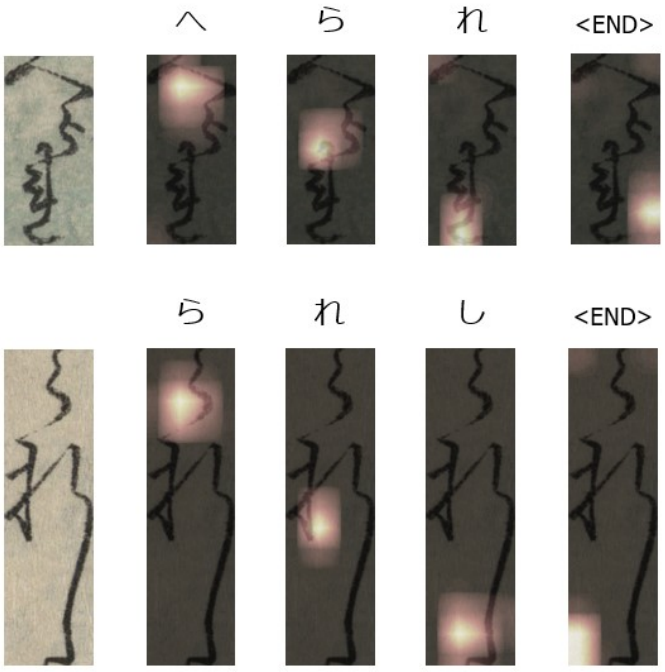


Figure 5.5. Visualization of the attention weights for single line text written vertically.

B. Multiple line recognition

Next, we evaluated the ARCED model for recognizing multiple text line images in the level 3 subset. This task is challenging because the model has to find the start of the first line and read all the characters in this line before finding the next line. Table 5.3 compares the recognition error rates by ARCED and the state-of-the-art methods reported in Section 4.4 on the test set of the level 3 subset. The ARCED model achieved CER of 12.69% and

SER of 58.58% on the test set, which reduced CER and SER to two-thirds compared with the best DCRN method combined with line segmentation. The results confirm that ARCED works well with the multiple text lines images and outperforms the state-of-the-art recognition accuracy in Section 4.4.

Table 5.3. Recognition error rates (%) on level 3 test set.

Model	**CER**	**SER**
2DBLSTM	46.73	98.81
CNN + 2DBLSTM	44.18	97.16
Seg + DCRN-o_12	26.70	82.57
Seg + End-to-End DCRN	18.50	73.70
ARCED	**12.69**	**58.58**

Again, this seems to be due to the attention-based seq2seq approach, the row-column BLSTM, and the residual LSTM. Moreover, it avoids errors due to line segmentation since ARCED does not need line segmentation.

Figure 5.6 shows a correctly recognized sample and its visualization of attention weights. We can see that the attention mechanism first focuses from the top right of the image to read the first character, shifts one character down to read the next character, and shift one more character to read the last character in the first line. Then, the attention mechanism returns to the top and shifts one line to the left to read the next line. This mechanism is again similar to us when we read vertical multiple text lines.

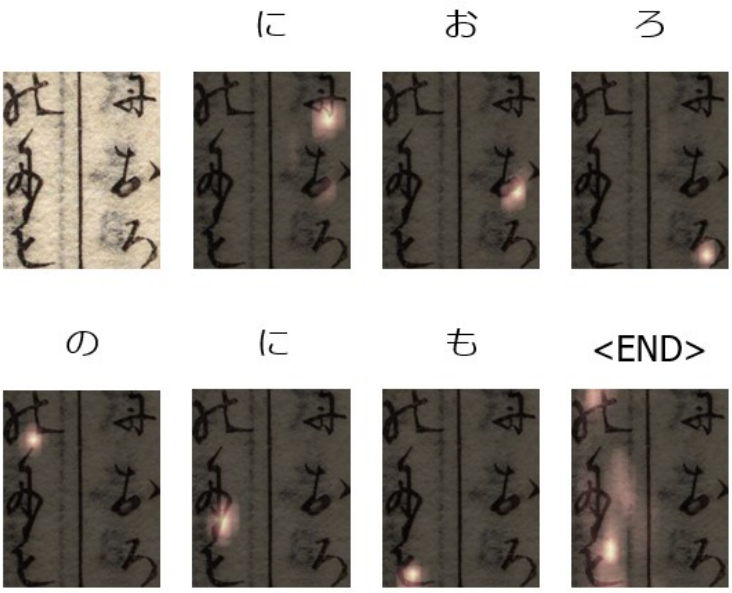


Figure 5.6. Visualization of the attention weights for multiple lines of text written vertically.

C. Row-column encoder

To verify the effect of the row-column BLSTM in the encoder, we prepared two variants. The first one is named Row-Encoder, which uses only the row BLSTM in the encoder. The second one is named Column-Encoder, which uses only the column BLSTM in the encoder. The other components of these models are the same as the ARCED model. Table 5.4 compares their recognition error rates.

Table 5.4. Recognition error rates (%) with different encoders.

Model	Level 2 test set		Level 3 test set	
	CER	SER	CER	SER
Row-Encoder	4.66	12.68	19.23	72.61
Column-Encoder	4.38	12.05	12.82	56.64
Row-Column Encoder (ARCED)	4.15	11.43	12.69	58.58

In both the subsets, the Column-Encoder model slightly outperforms the Row-Encoder model. Since text lines in Japanese historical documents are written vertically from top to bottom. The ARCED model outperformed on them. The results imply that the row-column BLSTM in the encoder improves the performance of the attention-based seq2seq model for the text recognition task in the Japanese historical documents. This seems to

be due to the row-column BLSTM that helps the encoder to capture the sequential order information in both the vertical and horizontal directions in the feature grid *F*.

D. Residual LSTM

To verify the effect of the residual LSTM network in the decoder, we prepared two variants. The first one is the same as the ARCED model except using the residual LSTM network, which is named ARCED_w/o_resLSTM. The second one is the same as ARCED except using the residual connection, which is named ARCED_w/o_resCon. Table 5.5 compares their recognition error rates with the ARCED model.

In both the level 2 and level 3 subsets, the ARCED model outperforms the ARCED_w/o_resLSTM model. The results show that the residual LSTM network between the attention vector and the softmax layer in the decoder improves the performance of the attention-based seq2seq model for the text recognition task in the Japanese historical documents. This seems to be due to the residual LSTM that helps the decoder to take advantage of all past attention vectors when it produces the predictive distributions. The ARCED model again outperforms ARCED_w/o_resCon in both the level 2 and level 3 subsets. The results show that the residual connection improves the performance of the ARCED model.

Table 5.5. Recognition error rates (%) with different decoders.

Model	Level 2 test set		Level 3 test set	
	CER	SER	CER	SER
ARCED_w/o_resCon	5.67	15.35	29.67	86.64
ARCED_w/o_resLSTM	4.53	12.43	14.43	62.54
ARCED	4.15	11.43	12.69	58.58

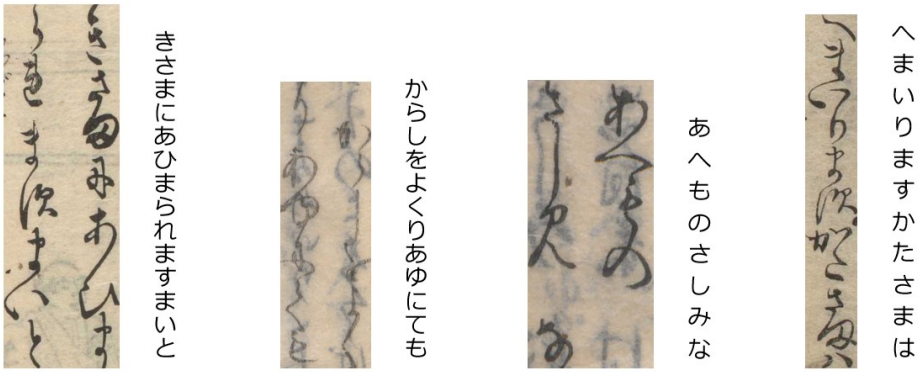
5.3.3. Analysis on recognized and misrecognized samples

Figure 5.7 shows some correctly recognized and misrecognized samples by the ARCED model in the level 3 dataset. The correctly recognized samples show that the ARCED model is effective in recognizing show-through patterns, connected patterns, and noisy patterns. For each misrecognized sample, the left image is the input, the middle text is the

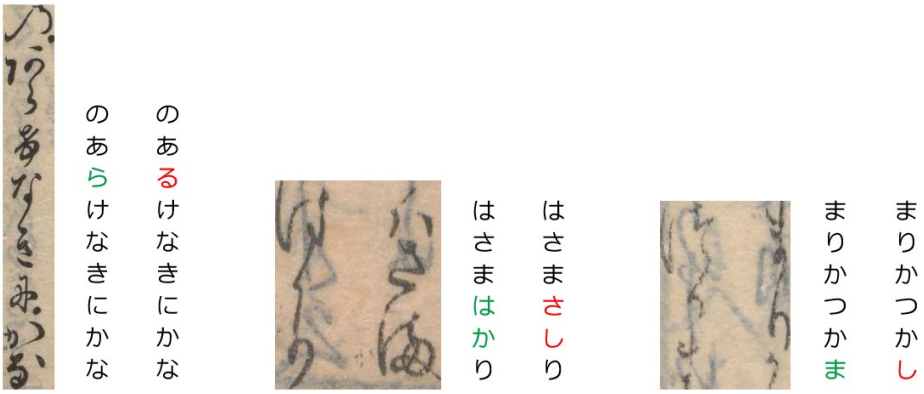
ground-truth, and the right text is the resulting recognition. Most of the misrecognized samples are missing only one or two characters in the ground-truth.

5.4. Conclusion.

In this chapter, we presented an attention-based row-column encoder-decoder model named ARCED for recognizing multiple text lines of deformed Kana sequences in Japanese historical documents. We introduced the row-column BLSTM in the encoder and the residual LSTM in the decoder. Following the experiments on the level 2 and level 3 subsets of the Kana_PRMU dataset, our proposed ARCED model achieved 4.15% and 12.69% character error rates in the test sets of level 2 and level 3, respectively. First, the attention based seq2seq approach can recognize both single and multiple text lines images, and results in a drastically reduced error rate compared to the previous state-of-the-art methods. Second, the row-column BLSTM in the encoder further reduced the error rate of the attention-based model by capturing the sequential order information in both the vertical and horizontal directions. Third, the residual LSTM further reduced the error rate by taking advantage of all the past attention vectors while generating the predictive distributions.



a). Correctly recognized samples in spite of show-through.



b). Misrecognized samples.

Figure 5.7. Correctly recognized and misrecognized samples.

Chapter 6. Conclusions and Future works

6.1. Conclusions

In this thesis, we presented a model of Deep Convolutional Recurrent Network (DCRN) for recognizing offline handwritten text lines without explicit segmentation of characters. The proposed DCRN model consists of three components: feature extractor by CNN, encoder by BLSTM, and decoder by CTC and has two approaches: pretrained CNN approach, and End-to-End approach. For decoding, we applied the CTC beam search combined with the tri-gram language model to obtain the final label sequence. The extensive experiments on standard benchmarks for offline handwritten Japanese text recognition show that the DCRN model outperforms the previous works of the segmentation-based method, and the tri-gram language model improves the performance of the DCRN model. We also propose an upgraded version of DCRN: Attention Augmented Convolutional Recurrent Network (AACRN) model which employs 1D self-attention mechanism in the encoder. The self-attention module is complementary to RNN in the encoder and helps the encoder to capture long-range and multi-level dependencies across an input sequence. The experiment results show that the AACRN model outperforms the DCRN model and the 1D self-attention mechanism improves the performance of the AACRN model.

Convolutional Neural Network (CNN) are successfully employed as a feature extractor to extract the visual features from an input image in the DCRN models. However, it processes the information in a local neighborhood, so that it might not extract information from long-distance locations in an input image. In this work, we propose an upgraded version of DCRN: 2D Self-Attention Convolutional Recurrent Network (2D-SACRN) which introduces a 2D self-attention mechanism in the feature extractor to help the CNN to capture the relationships between widely separated spatial regions in an input image. The extensive experiments on three widely used datasets: IAM Handwriting (English), Rimes (French), and TUAT Kondate (Japanese) show that the proposed model achieves similar or better accuracy when compared to state-of-the-art models in all datasets. Furthermore, the visualization of the 2D self-attention map shows that the 2D self-attention mechanism helps the feature extractor to capture the relationships between widely separated spatial regions and improves the performance of the 2D-SACRN model.

Deep Neural Networks typically require a large number of patterns per category for training. However, for many handwriting datasets, especially handwritten Japanese/Chinese datasets, the number of categories is large while the number of patterns per category is limited, so that it is necessary to apply a data argumentation method. In this work, we propose a synthetic pattern generation method which synthesize handwritten text line images from sentences in corpora and handwritten character patterns in the isolated character database with elastic distortions. The experiments on the offline handwritten Japanese text dataset show that the synthetic pattern generation method improves the performance of the DCRN model.

For the historical document recognition topic, we presented 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. We employed the DCRN model with two approaches for level 2. Then, we proposed a method of vertical text line segmentation and multiple line concatenation before applying DCRN for level 3. We also examined a two-dimensional BLSTM (2DBLSTM) based method for level 3. For level 2, the end-to-end approach achieved the best CER of 10.90% and SER of 27.70%. For level 3, the vertical text line segmentation and concatenation approach achieved the best CER of 12.30% and SER of 54.90% when trained by both the level 2 and level 3 datasets. Finally, we presented an attention-based row-column encoder-decoder model named ARCED for recognizing multiple text lines of deformed Kana sequences in Japanese historical documents. We introduced the row-column BLSTM in the encoder and the residual LSTM in the decoder. Following the experiments on the level 2 and level 3 subsets of the Kana_PRMU dataset, the proposed ARCED model achieved 4.15% and 12.69% character error rates in the test sets of level 2 and level 3, respectively. First, the attention based seq2seq approach can recognize both single and multiple text lines images, and results in a drastically reduced error rate compared to the previous state-of-the-art methods. Second, the row-column BLSTM in the encoder further reduced the error rate of the attention-based model by capturing the sequential order information in both the vertical and horizontal directions. Third, the residual LSTM further reduced the error rate by taking advantage of all the past attention vectors while generating the predictive distributions.

6.2. Future Works

For offline handwritten text recognition, however, some problems remain: the multi-line text data should be considered, and the character set of the Japanese handwritten recognition task should be extended to the JIS level 2 set. Firstly, we will improve our DCRN models further in terms of recognition accuracy for not only single line text images but also multi-line text images. For example, instead of using the BLSTM to build the encoder, we will try to use Multidimensional BLSTM to build the encoder. We also plan apply the attention-based model to recognize the multiple text line and compare with the Multidimensional BLSTM-based model.

Secondly, we will extend the character set to cover the JIS level 2 characters (about 7000 characters). The new databases will be used for training and evaluating the DCRN models. Finally, the language model made by RNN will be integrated with the system and compared with the tri-gram language model.

For the Japanese historical document recognition, the sequence error rate is so high that linguistic context must be incorporated. We will apply the language statistics and context processing to improve the accuracy of the systems. Tri-gram language model and RNN language model will be considered. This works only focused on the Kana characters, so we plan to accumulate more data including Kanji characters to improve the accuracy and recognize the whole character set.

We also will apply the proposed models and the synthetic pattern generation method for the other offline handwritten text benchmarks such as offline handwritten Vietnamese text recognition or offline handwritten Chinese text recognition.

References

1. Nguyen, K.C., Nguyen, C.T., Nakagawa, M.: A Segmentation Method of Single- and Multiple-Touching Characters in Offline Handwritten Japanese Text Recognition. IEICE Trans. Inf. Syst. E100.D, 2962–2972 (2017).
2. Qiu-Feng Wang, Fei Yin, Cheng-Lin Liu: Handwritten Chinese Text Recognition by Integrating Multiple Contexts. IEEE Trans. Pattern Anal. Mach. Intell. 34, 1469–1481 (2012).
3. El-Yacoubi, A., Gilloux, M., Sabourin, R., Suen, C.Y.: An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. IEEE Trans. Pattern Anal. Mach. Intell. 21, 752–760 (1999).
4. España-Boquera, S., Castro-Bleda, M.J., Gorbe-Moya, J., Zamora-Martinez, F.: Improving offline handwritten text recognition with hybrid HMM/ANN models. IEEE Trans. Pattern Anal. Mach. Intell. 33, 767–779 (2011).
5. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Trans. Pattern Anal. Mach. Intell. 31, 855–868 (2009).
6. Graves, A., Schmidhuber, J.: Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. Adv. Neural Inf. Process. Syst. 21, NIPS'21. 545–552 (2008).
7. Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In: Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR. pp. 285–290 (2014).
8. Shi, B., Bai, X., Yao, C.: An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Trans. Pattern Anal. Mach. Intell. 39, 2298–2304 (2017).
9. Bluche, T., Messina, R.: Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR. pp. 646–651 (2017).

10. Ly, N.-T., Nguyen, C.-T., Nguyen, K.-C., Nakagawa, M.: Deep Convolutional Recurrent Network for Segmentation-Free Offline Handwritten Japanese Text Recognition. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR). pp. 5–9 (2017).

11. Ly, N.T., Nguyen, C.T., Nakagawa, M.: Training an End-to-End Model for Offline Handwritten Japanese Text Recognition by Generated Synthetic Patterns. In: Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 74–79 (2018).

12. Ly, N.-T., Nguyen, K.-C., Nguyen, C.-T., Nakagawa, M.: Recognition of Anomalously Deformed Kana Sequences in Japanese Historical Documents. IEICE Trans. Inf. Syst. Vol.E102-D, (2019).

13. Puigcerver, J.: Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR. pp. 67–72 (2017).

14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5999–6009 (2017).

15. Dong, L., Xu, S., Xu, B.: Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. pp. 5884–5888 (2018).

16. Kitadai, A., Takakura, J., Ishikawa, M., Nakagawa, M., Baba, H., Watanabe, A.: Document Image Retrieval to Support Reading Mokkans. In: 2008 The Eighth IAPR International Workshop on Document Analysis Systems. pp. 533–538 (2008).

17. Terasawa, K., Shima, T., Kawashima, T.: A Fast Appearance-Based Full-Text Search Method for Historical Newspaper Images. In: 2011 International Conference on Document Analysis and Recognition. pp. 1379–1383 (2011).

18. Van Phan, T., Baba, H., Watanabe, A., Nakagawa, M.: A re-assembling scheme of fragmented Mokkan images. In: Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing - HIP ’13. p. 22 (2013).

19. Kitadai, A., Nakagawa, M., Baba, H., Watanabe, A.: Similarity Evaluation and Shape Feature Extraction for Character Pattern Retrieval to Support Reading Historical Documents. In: 2012 10th IAPR International Workshop on Document Analysis Systems. pp. 359–363 (2012).

20. PRMU: PRMU algorithm contest 2017, https://sites.google.com/view/alcon2017prmu/.

21. Nguyen, H.T., Ly, N.T., Nguyen, K.C., Nguyen, C.T., Nakagawa, M.: Attempts to recognize anomalously deformed Kana in Japanese historical documents. In: Proceedings of the 4th International Workshop on Historical Document Imaging and Processing - HIP2017. pp. 31–36 (2017).

22. Eskenazi, S., Gomez-Krämer, P., Ogier, J.-M.: A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognit. 64, 1–14 (2017).

23. Graves, A.: Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks. In: Guide to OCR for Arabic Scripts. pp. 297–313 (2012).

24. Luong, T., Pham, H., Manning, C.D.: Effective Approaches to Attention-based Neural Machine Translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1412–1421 (2015).

25. Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. (2014).

26. Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., Bengio, Y.: End-to-end attention-based large vocabulary speech recognition. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4945–4949 (2016).

27. Wang, C., Yin, F., Liu, C.-L.: Memory-Augmented Attention Model for Scene Text Recognition. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 62–67 (2018).

28. Chowdhury, A., Vig, L.: An Efficient End-to-End Neural Model for Handwritten Text Recognition. Br. Mach. Vis. Conf. 2018, BMVC 2018. (2018).

29. Bluche, T., Louradour, J., Messina, R.: Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR). pp. 1050–1055 (2017).

30. Ly, N.T., Nguyen, C.T., Nakagawa, M.: An attention-based row-column encoder-decoder model for text recognition in Japanese historical documents. Pattern Recognit. Lett. 136, 134–141 (2020).

31. Kim, M.S., Jang, M.D., Choi, H. Il, Rhee, T.H., Kim, J.H., Kwag, H.K.: Digitalizing Scheme of Handwritten Hanja Historical Documents. In: Proceedings - First International Workshop on Document Image Analysis for Libraries - DIAL 2004. pp. 321–327 (2004).

32. Van Phan, T., Cong Nguyen, K., Nakagawa, M.: A Nom historical document recognition system for digital archiving. Int. J. Doc. Anal. Recognit. 19, 49–64 (2016).

33. Kimura, F., Takashina, K., Tsuruoka, S., Miyake, Y.: Modified Quadratic Discriminant Functions and the Application to Chinese Character Recognition. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-9, 149–153 (1987).

34. Yang, H., Jin, L., Sun, J.: Recognition of Chinese text in historical documents with page-level annotations. In: Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR. pp. 199–204 (2018).

35. Valy, D., Verleysen, M., Chhun, S., Burie, J.-C.: Character and Text Recognition of Khmer Historical Palm Leaf Manuscripts. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 13–18 (2018).

36. IAM Aachen splits, https://www.openslr.org/56/, last accessed 2021/02/19.

37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015).

38. Xu, Y., Mo, T., Feng, Q., Zhong, P., Lai, M., Chang, E.I.-C.: Deep learning of feature representation with multiple instance learning for medical image analysis. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1626–1630 (2014).

39. Athiwaratkun, B., Kang, K.: Feature Representation in Convolutional Neural Networks. (2015).

40. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comput. 9, 1735–1780 (1997).

41. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016).

42. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the ACM International Conference Proceeding Series. pp. 369–376 (2006).

43. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning, ICML 2015. pp. 448–456 (2015).

44. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local Neural Networks. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 7794–7803 (2018).

45. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-Attention Generative Adversarial Networks. 36th Int. Conf. Mach. Learn. ICML 2019. 2019-June, 12744–12753 (2018).

46. Sueiras, J., Ruiz, V., Sanchez, A., Velez, J.F.: Offline continuous handwriting recognition using sequence to sequence neural networks. Neurocomputing. 289, 119–128 (2018).

47. Moysset, B., Messina, R.: Are 2D-LSTM really dead for offline text recognition? In: International Journal on Document Analysis and Recognition. pp. 193–208 (2019).

48. Bluche, T.: Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition. Neural Inf. Process. Syst. (2016).

49. Kang, L., Riba, P., Rusiñol, M., Fornés, A., Villegas, M.: Pay Attention to What You Read: Non-recurrent Handwritten Text-Line Recognition. arXiv. (2020).

50. Bluche, T.: Deep Neural Networks for Large Vocabulary Handwritten Text Recognition, https://tel.archives-ouvertes.fr/tel-01249405, (2015).

51. Poznanski, A., Wolf, L.: CNN-N-Gram for HandwritingWord Recognition. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 2305–2314 (2016).

52. Chen, B., Zhu, B., Nakagawa, M.: Training of an on-line handwritten Japanese character recognizer by artificial patterns. Pattern Recognit. Lett. 35, 178–185 (2014).

53. Leung, K.C., Leung, C.H.: Recognition of handwritten Chinese characters by combining regularization, Fisher's discriminant and distorted sample generation. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR. pp. 1026–1030 (2009).

54. Wigington, C., Stewart, S., Davis, B., Barrett, B., Price, B., Cohen, S.: Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR. pp. 639–645 (2017).

55. Nakagawa, M., Matsumoto, K.: Collection of on-line handwritten Japanese character pattern databases and their analyses. Int. J. Doc. Anal. Recognit. 7, 69–81 (2004).

56. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference. pp. 1724–1734 (2014).

57. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. In: Neural Networks. pp. 602–610 (2005).

58. Greff, K., Srivastava, R.K., Koutnik, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: A Search Space Odyssey. IEEE Trans. Neural Networks Learn. Syst. 28, 2222–2232 (2017).

59. Deng, Y., Kanervisto, A., Ling, J., Rush, A.M.: Image-to-markup generation with coarse-to-fine attention, https://dl.acm.org/citation.cfm?id=3305483, (2017).

60. Kim, J., El-Khamy, M., Lee, J.: Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition. (2017).

Author's Publications and Awards

Awards

1. Tokyo University of Agriculture and Technology, **TUAT President's Award for Students**, Mar 2018.
2. The Special Interest Group of Computers and the Humanities (Information Processing Society of Japan - IPSJ), "Recognizing anomalously deformed Kana by Deep Convolutional Recurrent Network", **PRMU CH Award**, Jan 2018.
3. The Special Interest Group of Pattern Recognition and Media Understanding (The Institute of Electronics, Information and Communication Engineers, Japan - IEICE), The 21st PRMU Algorithm Contest, **Best Algorithm Award**, Dec 2017.
4. The 4th International Workshop on Historical Document Imaging and Processing (HIP2017), **The IAPR Best Paper Award**, Nov 2017.

Journal Papers

1. **N. T. Ly**, K. C. Nguyen, C. T. Nguyen, and M. Nakagawa, "Recognition of Anomalously Deformed Kana Sequences in Japanese Historical Documents," IEICE Transactions on Information and Systems Vol.E102-D, No.8, pp.1554-1564, August 2019. (Chapter 4).
2. **N. T. Ly**, C. T. Nguyen, and M. Nakagawa, "An attention-based row-column encoder-decoder model for text recognition in Japanese Historical Documents," Pattern Recognition Letters, Vol. 136, pp. 134-141, August 2020. (Chapter 5).

International Conference Papers

1. **N. T. Ly**, H. T. Nguyen, and M. Nakagawa, "2D Self-Attention Convolutional Recurrent Network for Offline Handwritten Text Recognition," The 16th IAPR International Conference on Document Analysis and Recognition, 2021, pp. 191-204. (Chapter 3).

2. **N. T. Ly**, C. T. Nguyen, and M. Nakagawa, "Attention Augmented Convolutional Recurrent Network for Handwritten Japanese Text Recognition," 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2020, pp. 163-168. (Chapter 3).

3. **N. T. Ly**, L. Liu, C. Y. Suen and M. Nakagawa, "Hand-drawn Object detection for scoring Wartegg Zeichen Test", Second International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI 2020), Zhongshan City, China, 2020.

4. **N. T. Ly**, C. T. Nguyen, and M. Nakagawa, "An Attention-Based End-to-End Model for Multiple Text Lines Recognition in Japanese Historical Documents," 2019 International Conference on Document Analysis and Recognition (ICDAR), 2019, pp. 629-634. (Chapter 5).

5. **N. T. Ly**, C. T. Nguyen, and M. Nakagawa, "Training an End-to-End Model for Offline Handwritten Japanese Text Recognition by Generated Synthetic Patterns," 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2018, pp. 74-79. (Chapter 3).

6. **N. T. Ly**, C. T. Nguyen, K. C. Nguyen, and M. Nakagawa, "Deep Convolutional Recurrent Network for Segmentation-Free Offline Handwritten Japanese Text Recognition," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, pp. 5-9. (Chapter 3).

Joint Works

1. T. T. Ngo, H. T. Nguyen, **N. T. Ly**, and M. Nakagawa, "Recurrent neural network transducer for Japanese and Chinese offline handwritten text recognition," ICDAR 2021: Document Analysis and Recognition – ICDAR 2021 Workshops, pp. 364-376, 2021.

2. A. D. Le, D. Mochihashi, K. Masuda, H. Mima and **N. T. Ly**, "Recognition of Japanese historical text lines by an attention-based encoder-decoder and text line generation," 2019 5th International Workshop on Historical Document Imaging and Processing (HIP), 2019, pp.37-41.

3. H. T. Nguyen, **N. T. Ly**, K. C. Nguyen, C. T. Nguyen, and M. Nakagawa, "Attempts to recognize anomalously deformed Kana in Japanese historical documents", 2017

4th International Workshop on Historical Document Imaging and Processing (HIP), 2017, pp. 31-36.

4. H. D Nguyen, **N. T. Ly**, H. Truong, and D. D Nguyen, “Multi-Column CNNs for skeleton based human gesture recognition”, 2017 9th International Conference on Knowledge and Systems Engineering (KSE), 2017, pp. 179-184.

Domestic Conference Papers

1. 耒代誠仁、**Nam Tuan Ly**、Kha Cong Nguyen、中川正樹、山本和明、“階層化された情報システムのためのくずし字解読機能の試作、”日本情報考古学会第42回大会, 岡山大学津島キャンパス, 2019.

2. 佐藤 旭、小林 心、**Ly Nam Tuan**、Nguyen Tuan Cuong、北本 朝展、中川正樹、“日本古典籍くずし字文書の文字列認識、”第119回人文科学とコンピュータ研究会発表会、大阪市、02/2019.