Received September 8, 2020, accepted September 17, 2020, date of publication September 22, 2020, date of current version October 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3025769

Multi-Lingual Optical Character Recognition System Using the Reinforcement Learning of Character Segmenter

JAEWOO PARK[1], (Student Member, IEEE), EUNJI LEE[1], (Graduate Student Member, IEEE), YOONSIK KIM[1], ISAAC KANG[1], (Graduate Student Member, IEEE), HYUNG IL KOO[2], (Member, IEEE), AND NAM IK CHO[1], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, South Korea
[2]Department of Electrical and Computer Engineering, Ajou University, Suwon 16499, South Korea

Corresponding author: Hyung Il Koo (hikoo@ajou.ac.kr)

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and communications Technology Planning and Evaluation (IITP) under Grant IITP-2020-2020-0-01461.

ABSTRACT In this article, we present a new multi-lingual Optical Character Recognition (OCR) system for scanned documents. In the case of Latin characters, current open source systems such as Tesseract provide very high accuracy. However, the accuracy of the multi-lingual documents, including Asian characters, is usually lower than that for Latin-only documents. For example, when the document is the mix of English, Chinese and/or Korean characters, the OCR accuracy is lowered than English-only because the character/text properties of Chinese and Korean are quite different from Latin-type characters. To tackle these problems, we propose a new framework using three neural blocks (a segmenter, a switcher, and multiple recognizers) and the reinforcement learning of the segmenter: The segmenter partitions a given word image into multiple character images, the switcher assigns a recognizer for each sub-image, and the recognizers perform the recognition of assigned sub-images. The training of recognizers and switcher can be considered traditional image classification tasks and we train them with a supervised learning method. However, the supervised learning of the segmenter has two critical drawbacks: Its objective function is sub-optimal and its training requires a large amount of annotation efforts. Thus, by adopting the REINFORCE algorithm, we train the segmenter so as to optimize the overall performance, i.e., we minimize the edit distance of final recognition results. Experimental results have shown that the proposed method significantly improves the performance for multi-lingual scripts and large character set languages without using character boundary labels.

INDEX TERMS Deep learning, document analysis, optical character recognition.

I. INTRODUCTION

Optical Character Recognition (OCR) is an essential task for various applications, and numerous approaches have been considered so far. Recently, the most widely used OCR models are based on long short term memory-connectionist temporal classification (LSTM-CTC) [1] or sequence to sequence (seq2seq) models with attention [2], which takes a sequence of slices (narrow sub-images) as inputs and yields a sequence of recognition results. This approach allows us to build an OCR system without explicit character segmentation. Also, it naturally learns inter-character correlations by exploiting character-transition statistics. Therefore, many current open source systems such as Tesseract [3] are based on this LSTM-CTC model and provide the state-of-the-art performance in English recognition. However, the accuracy of LSTM-based methods for other languages such as Chinese and Korean are not as high as English. This is mainly because these kinds of languages have a large number of characters compared to English; Chinese and Korean have thousands of character classes and there are a huge number of possible cases even if we only consider pairwise combinations.

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Sharif.

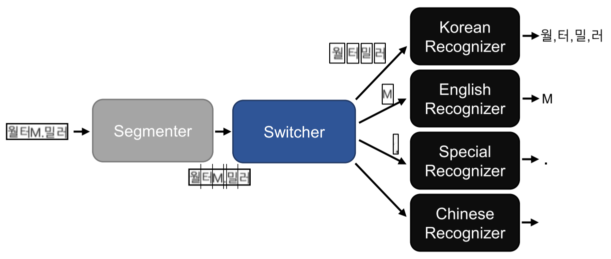

FIGURE 1. **The overall framework of the proposed OCR system: The segmenter partitions a word image into character images, the switcher assigns a recognizer to each character image, and the recognizers perform the recognition of assigned images.**

Therefore, unlike English, explicit character segmentation and individual character recognition are important for these types of languages.

Based on these observations, we propose a new multi-lingual OCR framework: We develop recognizers for individual languages and incorporate them into a single framework in a globally optimized way. Precisely, we use three neural blocks (a segmenter, a switcher, and multiple recognizers) as in Figure 1. First, the segmenter partitions a given word image into character images. Then, for each character image, the switcher assigns a recognizer that is responsible for the recognition of corresponding characters. Finally, each reconginzer performs the character recognition of assigned sub-images. Although some characters in different languages may have similar shapes, the confusion in the switcher is not common because each language has distinctive properties such as intra/inter spacings and strokes as shown in Figure 2. Also, although the segmenter and switcher provide character-level decisions, recognizer can perform word-level recognition by concatenating character images into a word. In this way, we are able to fully exploit the characteristic of individual languages in a scalable way.

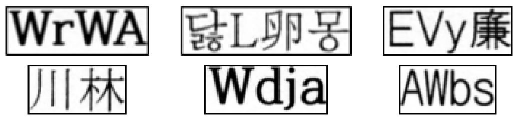

FIGURE 2. **Word image examples. Character segmentation is not straightforward due to overlapping characters, large intra-character spaces, and so on.**

The role of the proposed segmenter block may seem to be the same to that of conventional character segmentation. Actually, numerous OCR methods were based on character-level segmentation and many character segmentation methods were proposed. For example, projection profiling based methods [4]–[8] focused on spacing between characters and researchers developed ad-hoc rules to identify them. However, there are many exceptions: Spacing can occur within single characters, neighboring characters can touch each other, and spacing between characters may not be clear due to geometric and photometric distortions as shown in Figure 2. To address these challenges, deep neural network-based character segmentation methods were recently proposed [9], [10]. However, this supervised approach requires a large amount of character boundary annotations, which complicates the deployment of this system to a range of image acquisition environments.

We believe that the role of the segmenter is not just to re-produce the ground truth partitions but to find a partition that maximizes the performance of the whole system. Notably, making the ground truth character boundaries is ambiguous as illustrated in Figure 2. When there is a large inter-character spacing, there are many possible boundary candidates. Also, adjacent characters may not be separated due to vertical overlapping. Therefore, we train the segmenter block so as to optimize the edit distance (Levenshtein distance) [11] of final recognition results in the reinforcement learning framework. This approach [12] not only allows us to have a globally optimized OCR system, but also minimizes annotation efforts since building a realistic word image dataset with character boundary annotations is an expensive task. Experimental results have showed that the proposed OCR system yields excellent performance for a large character set languages such as Chinese and Korean, and their mixed scripts.

II. RELATED WORK

Recent OCR researches are basically twofold: One is scene text recognition that deals with text in natural scene images [13]–[17] and the other is document OCR that recognizes text in document images [10], [18]–[20]. In document OCR, word localization is relatively easy, since characters are well-aligned and backgrounds are usually homogeneous. However, dictionary size in document images is large and high throughput is also required in the test phase. In this article, we focus on document image word recognition and discuss two standard approaches in this field as illustrated in Figure 3.

A. EXPLICIT SEGMENTATION-BASED APPROACH

The most straightforward approach to explicit character segmentation is a projection-profile-based method, which identifies the boundaries of characters with vertical projection profiles [4]–[8]. To reduce the difference between projection profiles patterns and character boundaries, algorithms such as adaptive threshold or morphology-based methods were usually employed in this approach [18], [21]–[25]. However, in spite of these ad-hoc methods, projection profile based methods suffered from under-segmentation (touching characters) and over-segmentation (spacing within single characters) [26]–[28].

To alleviate these problems, researchers attempted to apply deep learning to character segmentation [9], [10]. For instance, Zheng *et al.* [10] formulated character segmentation as a binary segmentation problem and used Fully Convolutional Network (FCN) to find character boundaries. Chernyshova *et al.* [9] suggested a lightweight word image

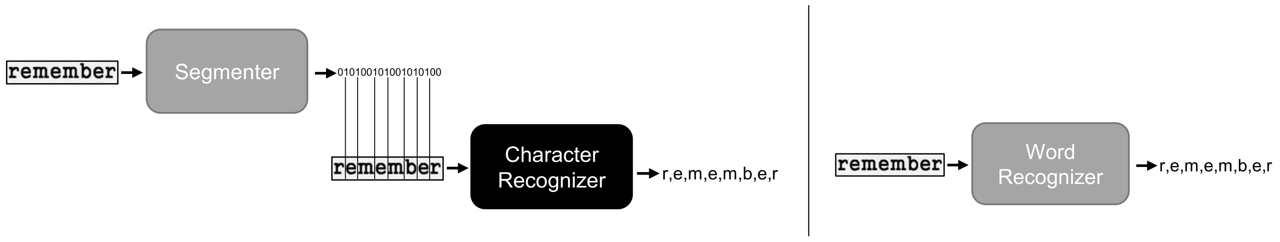

FIGURE 3. Two typical approaches for OCR. Left: Explicit segmentation-based OCR method and Right: Implicit segmentation-based OCR method.

segmentation model, which used dynamic programming to select the most probable boundaries in images. However, joint optimization of character segmentation and recognition has not been addressed and character segmentation blocks were trained in a sub-optimal way.

B. CHARACTER RECOGNITION WITHOUT EXPLICIT SEGMENTATION

LSTM based methods take word images as inputs and perform recognition without the explicit character segmentation as in Figure 3. For instance, LSTM-CTC methods used neural networks consisting of three types of layers: convolution layers, LSTM layers, and CTC layers. The convolution layers extract features from slices of word images, and LSTM layers yield the character classes of input features. Since multiple slices are made from a single character, the LSTM layers usually produce the same symbol multiple times, and the CTC layer suppressed them [1]. Seq2seq with attention model [2] used the encoder-decoder framework. Here, the encoder scans word images along the horizontal direction to make a single feature vector describing a whole input image, and the decoder yields character sequences by decoding the feature vector with the attention mechanism.

Since character boundary labels are not needed in the training of LSTM based models, these methods can be applied to a range of environments. Also, its objective function is global. Note that in the explicit character segmentation approach independent loss functions should be used for the training of individual sub-block. However, the approach of using *narrow slices* is not suitable for languages having a large number of characters. That is, each slice provides only a limited view of an entire character and many slices from different characters may look similar because of 2D juxtaposition of a relatively small number of glyphs. Hence, the accuracy for Chinese and Korean are not as high as English, and it gets worse for mixed scripts.

III. PROPOSED METHOD

The explicit-segmentation-based approach allows us to deal with a large number of characters and the implicit-segmentation-based approach enables the use a global objective function only with word-level annotations. In order to exploit the advantages of both approaches, we propose a new framework consisting of three blocks as shown in Figure 1. In this section, we first present the overview of the proposed system, and then, focus on the presentation of the segmenter block.

A. OVERVIEW

We consider an input word image $X \in \mathbb{R}^{H\times W}$ as a horizontal concatenation of character images, and our goal is to get a sequence of character labels in X. First, a segmenter $S(\cdot)$ takes a word image X as input and yields the boundary of characters

$$S(X) = (a_1, a_2, \cdots, a_W) \in \{0, 1\}^W \tag{1}$$

where 1s stand for character boundaries and 0s for others as in Figure 3. For a given character boundary map $a \in \{0, 1\}^W$, we can partition X into $|a| + 1$ sub-images, where $|a|$ is the number of 1s in a. For each sub-image, the switcher $L(\cdot)$ assigns a recognizer that is responsible for the recognition of that sub-image, and recognizers of each language $R_i(\cdot) \in \{R_1(\cdot), R_2(\cdot), \ldots, R_M(\cdot)\}$ finds the labels of assigned sub-images.

We train $L(\cdot)$ with character images from multiple languages, and $R_i(\cdot)$ with character images from corresponding language with a cross-entropy based objective function. Then, by using the trained switcher and recognizers in reward-evaluation, we train a segmenter $S(\cdot)$. We denote the recognition result of X with a character boundary map a as $Y(X, a)$, therefore, the recognition result of X using segmenter $S(\cdot)$ will be $Y(X, S(X))$.

B. SEGMENTER MODELING

For the reinforcement learning, we model the segmenter in a probabilistic way, rather than formulating it as a hard binary classification problem. To be precise, we perform the modeling of the policy distribution $\pi(a|X; \theta)$ that yields actions $a \in \{0, 1\}^W$ for a given X. We represent $\pi(a|X; \theta)$ by using an output of a network

$$f(X; \theta) \tag{2}$$

where θ is a set of parameters in the network.

However, unlike typical reinforcement learning situations, our action is a high-dimensional random vector (where individual random variables are strongly correlated). For example, when $|i - j|$ is small, it is very unlikely to have $a_i = a_j = 1$. This makes the element-wise sampling of action very inefficient, therefore, we add a post-processing block to the network output, so that we obtain $g(X;\theta) = (p_1, p_2, \cdots, p_W)$ satisfying

$$\pi(a|X;\theta) \simeq \prod_{i=1}^{W} p_i^{a_i}(1-p_i)^{1-a_i} \tag{3}$$

where $a = (a_1, a_2, \cdots, a_W)$.

To be precise, we apply three operations to $f(X;\theta)$ in order to get $g(X;\theta)$. First, we apply non-max suppression to $f(X;\theta)$ with a window size of 5 so that the sampling from (3) yields plausible cases (otherwise, $a_i = a_j = 1$ for small $|i-j|$ will commonly occur). Second, in order to pose a probabilistic sampling even for very strong candidates, we clip the probability with $\tau = 0.99$. Finally, we adopt an element-wise threshold operation to filter out low probability candidates at the early stage of training:

$$g(X;\theta) = \begin{cases} \min(\text{nonmax}(f(X;\theta)), \tau) & f(X;\theta) \geq \eta \\ 0 & f(X;\theta) < \eta \end{cases} \tag{4}$$

where η is a threshold value.

During the training phase, we draw actions from the policy distribution

$$a^{(j)} \sim \pi(X;\theta) \tag{5}$$

and evaluate their rewards based on edit distance of final results where j is a sample index. On the other hand, in the test phase, we select the most probable action:

$$S(X) = \arg\max_{a} \pi(a|X;\theta), \tag{6}$$

i.e., $a_i = 1$ only when $p_i \geq 0.5$.

C. REINFORCEMENT LEARNING OF SEGMENTER

We aim to minimize the edit distance derived from the segmenter, and we model the reward function as

$$r(X,a) = 1 - \frac{d(Y(X,a), Y)}{\max(|a|+1, N_Y)} \tag{7}$$

where $d(\cdot,\cdot)$ is an edit distance, Y is the ground truth word annotation of X, and N_Y is the number of characters in Y. The second term in (7) can be considered a length-normalized edit distance. Based on the reward, the objective function of our segmenter is formulated as

$$\begin{aligned} \mathcal{L}(\theta) &= -\mathbb{E}_a[r(X,a)] \\ &\simeq -\sum_j \pi(a^{(j)}|X,\theta) r(X,a^{(j)}). \end{aligned} \tag{8}$$

To apply the policy gradient method, we derive the gradient of the function (8) as

$$\begin{aligned} \nabla_\theta \mathcal{L} &= -\sum_j r(X,a^{(j)}) \nabla_\theta \pi(a^{(j)}|X,\theta) \\ &= -\sum_j r(X,a^{(j)}) \pi(a^{(j)}|X,\theta) \nabla_\theta \log\left(\pi(a^{(j)}|X,\theta)\right) \\ &= -\sum_j r(X,a^{(j)}) \pi(\cdot) \nabla_\theta \log\left(\prod_{i=1}^{W} p_i^{a_i^{(j)}} (1-p_i)^{1-a_i^{(j)}}\right) \\ &= -\sum_i \nabla_\theta \log(p_i) A_i + \nabla_\theta(\log(1-p_i)) B_i \\ &= -\nabla_\theta \log(f(X;\theta))^\top A - \nabla_\theta \log(1-f(X;\theta))^\top B \end{aligned} \tag{9}$$

where

$$A_i = \sum_j a_i^{(j)} \pi(a^{(j)}|X,\theta) r(X,a^{(j)}) \tag{10}$$

$$B_i = \sum_j (1-a_i^{(j)}) \pi(a^{(j)}|X,\theta) r(X,a^{(j)}), \tag{11}$$

$A = [A_1, \cdots, A_W]$ and $B = [B_1, \cdots, B_W]$. Also, we use the baseline method [29] to stabilize the learning by reducing variances in the gradient estimation. Since (6) is independent of sampled actions and the corresponding reward reflects the behavior of sampled actions, we use (6) as a baseline action, i.e., $b(X) = S(X)$:

$$\dot{A} = \sum_j a^{(j)} \pi(a^{(j)}|X,\theta)(r(X,a^{(j)}) - r(X,b(X))) \tag{12}$$

$$\dot{B} = \sum_j (1-a^{(j)}) \pi(a^{(j)}|X,\theta)(r(X,a^{(j)}) - r(X,b(X))). \tag{13}$$

Finally, we add a regularization term:

$$\begin{aligned} \nabla_\theta \mathcal{L} \simeq &-\nabla_\theta \log(f(X;\theta))^\top \dot{A} \\ &- \nabla_\theta \log(1-f(X;\theta))^\top \dot{B} - \lambda \nabla_\theta \|\theta\|^2 \end{aligned} \tag{14}$$

where λ is a hyper-parameter.

The overall process of learning segmenter is illustrated in Figure 4. Similar to [29], we use the REINFORCE algorithm to optimize the reward, rather than developing the whole reinforcement framework.

D. SEGMENTER TRAINING DETAILS

To help reliable training, we first train the segmenter using pseudo labels from the projection profile method [30] using an element-wise binary cross-entropy loss. Note that we do not need ground truth labels in this pre-training. Although the segmenter learned from projection profile methods shows poor performance, this pre-training provides a good initial point by assigning relatively high probabilities to candidates of character boundaries.

Also, we adopt an incremental learning scheme that gradually decreases η in (4). When all possible actions are allowed at the early stage of training, it might take longer training time to learn meaningful rules. Rather, by scheduling η, we provide promising candidates at the early stage, and gradually provide less promising ones to boundary candidates. We start η at 0.4 and linearly decrease it by 0.015 every epoch up to 20 epochs.

IV. DATASET BUILDING AND NEURAL NETWORK ARCHITECTURES

Datasets we use for the experiment are

- synthetic word image dataset for the training of a segmenter block,

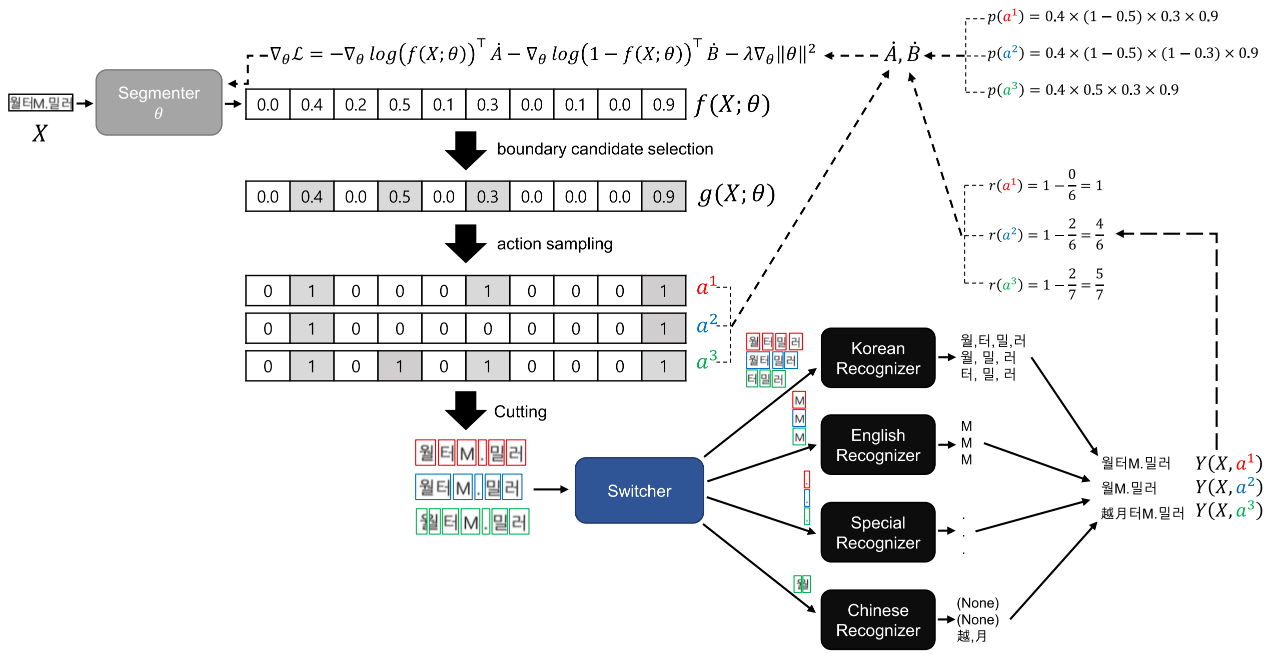

FIGURE 4. **Overview of the proposed method. In this figure, the overall steps are illustrated (sampling actions, reward evaluation, and $\dot{A}, \dot{B}$ computations). See the text for details.**

- real word image dataset for the testing of a segmenter block, and
- real character image dataset for the training/testing of switcher/recognizer blocks.

After explaining three datasets, the detailed structure of each neural block is followed. Although there can be an unlimited combination of different languages, we focus on the combination of English and two Asian characters, namely Chinese and Korean, as a specific implementation example.

A. SYNTHETIC WORD IMAGE DATASET

To the best of our knowledge, there is no publicly available word image dataset having mixed scripts with Korean, Chinese, and English. Hence, we build a synthetic multilingual word image training dataset. To this end, we first build our dictionaries by collecting words from Wikipedia. Then, we augment the dictionary by generating words with the random combinations of English (52 classes), Chinese (1, 817 classes), Korean (2, 350 classes), and special characters (46 classes) such as exclamation mark, rest, full stop, and brackets. Here, the character classes for Chinese and Korean are selected based on the appearance frequency in Wikipedia and the national language standards.

With these dictionaries, we create synthetic word images with five types of fonts and eight different scales. To make synthesized word images similar to real scanned images, we add compression artifacts and apply random rotation in $[-2°, 2°]$. Some images are shown in Figure 5. The numbers of synthetic word images are 208, 480 for Chinese, 314, 880 for Korean, and 325, 360 for English, respectively.

B. TEST WORD DATASET

For realistic test images, we have scanned printed documents with flatbed-scanners, and applied the following procedures to the scanned images: skew correction [31], text line extraction [32], and word segmentation [30]. In building the testset, we have used multi-script words by concatenating English, Chinese, and Korean characters as shown in Figure 5.

C. REAL CHARACTER IMAGE DATASET

In order to train switcher and recognizers, we collect real scanned character images as shown in Figure 6. Similar to the word dataset, we generate them with five types of fonts and eight different scales. Also, in order to cover various types of fonts and character styles that are not in the training dataset, we additionally collect character images from magazines and textbooks by cropping characters in scanned images.

D. NETWORK ARCHITECTURES

We use one segmenter, one switcher, and four recognizers and the details of the architectures will be discussed in this section. The architectures of each language recognizer used in this article are summarized in TABLE 1, and the architectures of character language switcher and segmenter are described in TABLE 2, where Conv$(c, i \times j, k \times m)$ means a convolution layer having c channels, a $(i \times j)$-sized kernel, k vertical strides, and m horizontal strides. BiLSTM(d) means a bidirectional LSTM layer with a dimension of d. MaxPool$(i \times j, k \times m)$ and AvgPool$(i \times j, k \times m)$ indicate max pooling and average pooling layer with $(i \times j)$-sized kernel with $k \times m$ strides, respectively. We use ReLU as activation function

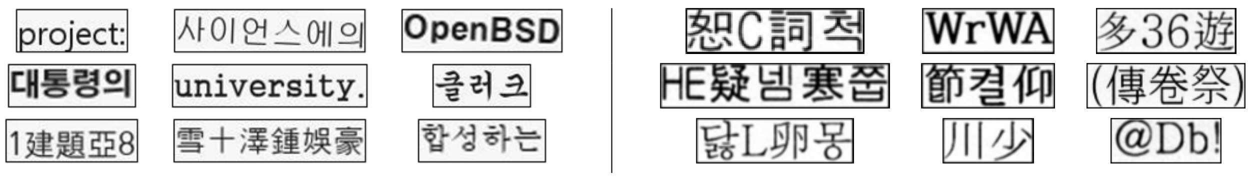

FIGURE 5. Examples of word images in our dataset. Left: Synthetically generated word images and Right: Scanned word images.

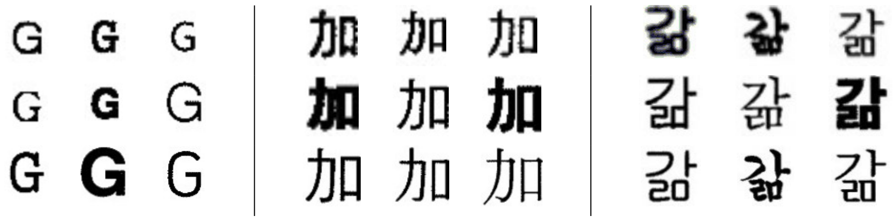

FIGURE 6. Examples of character images in our dataset. Left: Scanned English character images, Middle: Scanned Chinese character images and Right: Scanned Korean character images.

TABLE 1. Architectures of Recognizer of English (R_{eng}), Special (R_{spe}), Chinese (R_{chi}), and Korean (R_{kor}).

R_{eng}	R_{spe}	R_{chi}	R_{kor}		
Conv(6, 5×5, 1×1)	Conv(6, 5×5, 1×1)	Conv(12, 5×5, 1×1)	Conv(18, 5×5, 1×1)		
AvgPool(2×2, 2×2)	AvgPool(2×2, 2×2)	AvgPool(2×2, 2×2)	AvgPool(2×2, 2×2)		
Conv(16, 5×5, 1×1)	Conv(16, 5×5, 1×1)	Conv(32, 5×5, 1×1)	Conv(48, 5×5, 1×1)		
BatchNorm	BatchNorm	BatchNorm	BatchNorm		
AvgPool(2×2, 2×2)	AvgPool(2×2, 2×2)	AvgPool(2×2, 2×2)	AvgPool(2×2, 2×2)		
Conv(120, 5×8, 1×1)	Conv(120, 5×8, 1×1)	Conv(240, 5×8, 1×1)	Conv(360, 5×8, 1×1)		
BatchNorm	BatchNorm	BatchNorm	BatchNorm		
FC(84)	FC(84)	FC(168)	FC(84)	FC(84)	FC(84)
FC(53)	FC(47)	FC(1818)	FC(20)	FC(22)	FC(29)
Softmax	Softmax	Softmax	Softmax	Softmax	Softmax

for convolution layers and denote the batch normalization layer and the softmax function as BatchNorm and Softmax, respectively.

Since the character segmenter in [9] showed good performance with lightweight architectures, we design convolution layers of our segmenter based on their architecture. However, we modify strides and padding sizes so that the output feature map becomes $W \times 1$ from $W \times 32$ inputs at the end of convolution layers. Then, we use two layers of bidirectional LSTM on the top of them, in order to exploit entire word images more effectively. Although the segmenter takes word images as inputs, it only addresses a column-wise binary classification problem and needs a relatively small number of parameters. The switcher $L(\cdot)$ needs to capture shape features in a range of scripts, and we adopt the VGG-13 as a base model [33], which has a larger capacity than LeNet-5. However, to improve the efficiency, we reduce the number of channels and layers.

In order to exploit the characteristics of each language, we design the architectures of recognizers according to languages. For English and special character recognizers (R_{eng}, R_{spe}), we use LeNet-5 [34] based architectures with batch normalization and a fully connected layer, consisting of 53 and 47 classes respectively including an additional

TABLE 2. Architectures of Segmenter (S) and Switcher (L).

S	L
Conv(6, 5×5, 2×1)	Conv(32, 3×3, 1×1)
Conv(6, 5×5, 2×1)	MaxPool(2×2, 2×2)
Conv(6, 7×3, 2×1)	Conv(64, 3×3, 1×1)
Conv(6, 5×5, 2×1)	BatchNorm
Conv(6, 3×3, 2×1)	MaxPool(2×2, 2×2)
BiLSTM(8)	Conv(128, 3×3, 1×1)
BiLSTM(8)	BatchNorm
FC(1)	MaxPool(2×2, 2×2)
Sigmoid	Conv(256, 3×3, 1×1)
	BatchNorm
	MaxPool(2×2, 2×2)
	Conv(512, 3×3, 1×1)
	BatchNorm
	FC(1024)
	FC(256)
	FC(4)
	Softmax

Not-Recognizable image (NR) class. For the recognition of Chinese, which has a lot of classes and sophisticated patterns, we design the R_{chi} to have twice as many channels as R_{eng} with 1818 classes (including a NR class). In the case of Korean characters, we do not directly classify each character,

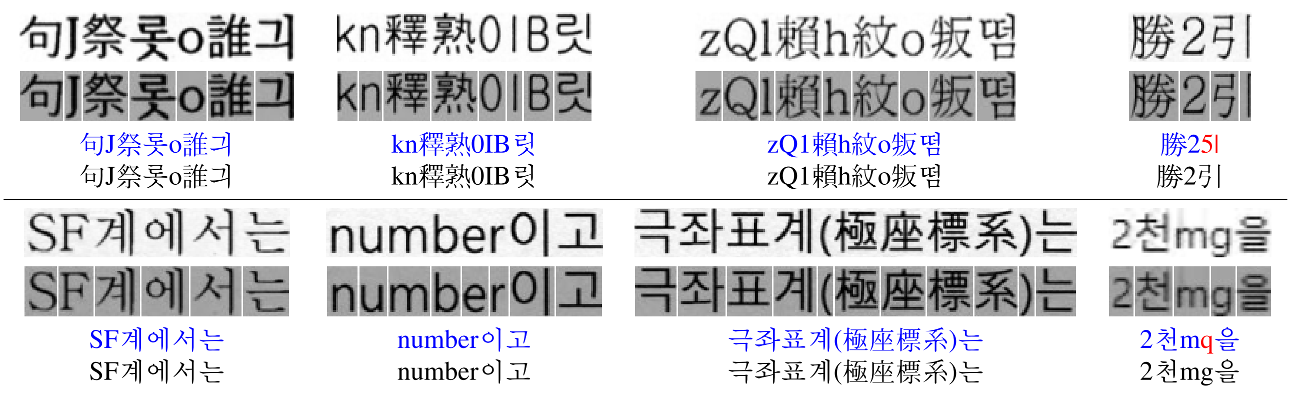

FIGURE 7. Examples of boundary predictions and OCR results of the proposed method on random sequences (upper block) and practical mixed examples (lower block). First row: Inputs (scanned word images), Second row: Segmenter prediction results, Third row: OCR results (where correct predictions are in blue color and wrong predictions in red color), and Last row: ground truth labels.

TABLE 3. Accuracy on multi-lingual documents (%). We denote testset words consisting of Chinese characters as Chi, English as Eng, and Korean as Kor. Plus sign '+' means mixed scripts. "Boundary supervised" means the training requires character boundary annotations. On the other hand, "Word supervised" means that the training requires word labels.

	Boundary supervised	Word supervised	Chi	Eng	Kor	Chi+Eng	Chi+Kor	Eng+Kor	Chi+Eng+Kor	Average
Tesseract [3]		✓	66.00	**81.21**	39.46	30.06	25.62	19.73	5.34	38.20
Projection profile [30]			77.33	18.59	48.09	58.75	51.59	38.78	47.37	48.64
Mixed [10]	✓		25.66	64.19	49.15	53.60	32.84	67.93	59.35	50.39
TwoStep [9]	✓		82.23	48.63	65.74	73.16	78.56	67.35	71.40	69.58
Proposed method		✓	**94.74**	77.01	**97.07**	**87.23**	**97.10**	**87.46**	**90.87**	**90.21**

since Korean characters consist of basic elements with fewer numbers of classes. Precisely, each Korean character can be split into three basic elements: *choseong* that comes at the head of a character, *jungseong* in the middle and/or right, and *jongseong* at the bottom. We use an additional NR class for *choseong*, *jungseong*, and *jongseong* respectively, and the final numbers of classes for them are 20, 22, and 29, which is much smaller than the size of the whole character set. Hence, we predict characters label by identifying the labels of these three elements at the same time, without character partitioning. Note that *jongseong* is not required in some characters and 29 classes consist of (a) 27 *jongseong* classes, (b) one case representing that *jongseong* is not used, and (c) one for the NR class. To recognize three elements, we use a three-headed-model. That is, Korean recognizer extracts features using a shared block (convolution layers), and then three independent fully connected blocks estimate the labels of three elements, respectively. We use three times as many as channels used in LeNet-5 for convolution layers and each fully connected layer has the same size to LeNet.

V. EXPERIMENTAL RESULTS

For the evaluation of the proposed method, we conduct extensive experiments. In experiments, we linearly resize word/character images to 32 pixels in height. For switcher/recognizers, we pad zero columns to make 44 × 32 images (44 is the maximum width of characters in our sets).

TABLE 4. The number of failures for Chi+Eng+Kor inputs having 1161 characters. We can see that (a) given correct character segmentation, other blocks (L, R_{eng}, R_{spe}, R_{chi}, R_{kor}) work well and (b) the segmenter (S) plays a key role for the performance.

	S	L	R_{eng}	R_{spe}	R_{chi}	R_{kor}	total
Projection profile	606	3	0	0	1	1	611
Mixed	449	18	2	0	3	0	472
Twostep	125	18	2	0	1	1	147
Proposed method	74	24	2	1	3	2	106

Since a trained switcher and recognizers are required to train our segmenter through reinforcement learning, we first train the switcher and recognizers. The switcher and character recognizers are trained for 100 epochs with Adam optimizer [35], where the learning rate is 0.0001 with a decay rate of 0.9 per ten epochs. As mentioned in Sec. III-D, we pre-trained the segmenter $S(\cdot)$ with pseudo labels for ten epochs with a learning rate of 0.0001, which is decreased by ten times at every three epochs. Then, we train $S(\cdot)$ with (14) for 20 epochs with a learning rate of 0.0001, which is decreased by ten times at every seven epochs. We set λ in (14) to 0.00001.

A. EVALUATION

We visualize some of our results in Figure 7. As shown, the proposed method can recognize multilingual word images successfully for both random character sequences and practical mixed examples. We compare the accuracy of our model

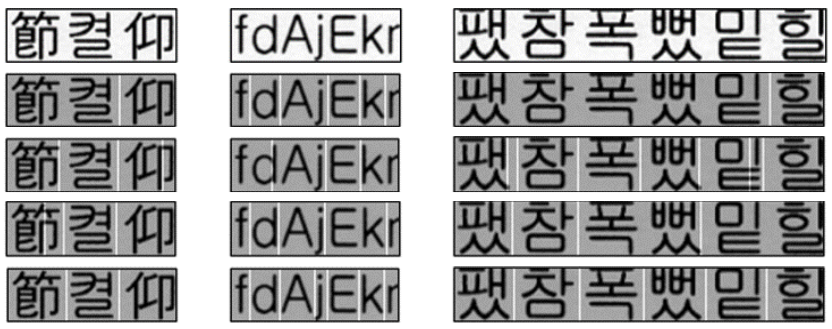

FIGURE 8. Comparison between character segmentation methods. First row: input word image examples, Second row: predicted boundary by [30], Third row: results of [10], Fourth row: results of [9], and Last row: results of our method.

TABLE 5. Ablation study of our proposed method. (%). We denote testset words language same as Table 3. "REINFORCE" means whether the model is trained using reinforcement learning and "η" means the value of η in training phase.

	REINFORCE	η	Chi	Eng	Kor	Chi+Eng	Chi+Kor	Eng+Kor	Chi+Eng+Kor	Average
Projection profile			77.33	18.59	48.09	58.75	51.59	38.78	47.37	48.64
Proposed method			58.11	19.86	43.28	44.47	57.40	39.65	40.74	43.36
Proposed method	✓	0.1	54.94	67.71	95.52	64.45	86.77	81.92	77.09	75.49
Proposed method	✓	0.4	89.39	**77.59**	95.77	86.34	95.19	**87.56**	90.44	88.90
Proposed method	✓	scheduling	**94.74**	77.01	**97.07**	**87.23**	**97.10**	87.46	**90.87**	**90.21**

with the conventional character segmentation-based OCR systems as well as Tesseract [3]. As summarized in TABLE 3, Tesseract shows superior performance on English character recognition. Since English has many similar characters (e.g., 'l' and 'I'), which should be identified based on the context, OCR system based on word-level processing (e.g., Tesseract using LSTM-CTC model) shows better performance than character segmentation-based methods. On the other hand, Tesseract does not have character level granularity, and performance is very low for mixed-scripts words (lower than the projection profile based method) [30]. Also, Tesseract shows poor performance for languages with a large number of characters such as Chinese and Korean.

We also evaluate the performance of other character segmentation-based OCR methods (neural networks based methods [9], [10] and a projection profile-based method [30]). We train two neural network-based methods with our dataset, and use the same recognizers and switcher for the fair comparison. Note that Twostep [9] and Mixed [10] methods require the ground-truth labels of character boundaries, while we do not need these ground truth character boundaries due to the reinforcement learning framework. As shown, they show poor performance compared to our method in most cases. Since [9] and [10] converge to small loss value on the synthetically generated word images, we believe that this result indicates the advantages of our global objective function that directly reflects the qualities of final results.

Figure 8. shows some character segmentation results, where predicted boundaries are shown in white lines. Unlike other character segmentation based methods, our method can detect the proper character boundaries for a range of situations and across script types.

TABLE 6. Architecture of Switcher-Segmenter (SS).

SS	
Conv(32, 5×5, 2×1)	
Conv(64, 5×5, 2×1)	
Conv(64, 7×3, 2×1)	
Conv(128, 5×5, 2×1)	
Conv(128, 3×3, 2×1)	
BiLSTM(32)	
BiLSTM(32)	
FC(1)	Conv(16, 1×3, 1×1)
Sigmoid	Conv(32, 1×3, 1×1)
	Conv(64, 1×3, 1×1)
	Conv(128, 1×3, 1×1)
	Conv(256, 1×3, 1×1)
	FC(512)
	FC(128)
	FC(5)
	Softmax
Segmenter output	Switcher output

Also, to clarify the contribution of segmenter, we compare the number of failure cases of individual modules. In the experiments, we use the same switcher and recognizers to focus on segmentation methods and use Chi+Eng+Kor word images to evaluate the performance on multiple languages. As shown in Table 4, errors in segmentation blocks are dominant, showing that the segmenter is a key component for the performance. By improving segmentation performance, our method outperforms other methods.

B. ABLATION STUDY

To evaluate the contribution of training schemes, we conduct additional experiments. First, we evaluate the performance by training the proposed segmenter only to mimic projection profile results, i.e., no reinforcement learning is applied. As

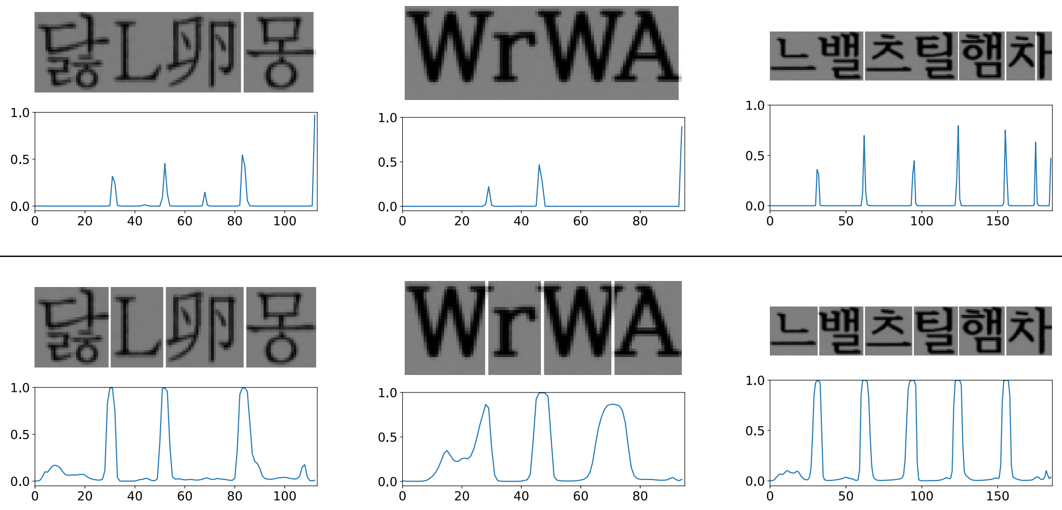

FIGURE 9. Effects of the reinforcement learning: the top row shows the results without the reinforcement and the bottom row are results with the reinforcement learning (including η-scheduling scheme). Blue graphs represent $f(X;\theta)$ values. Note that since we apply non-max suppression to obtain character boundaries ($g(X;\theta)$), only local maximum locations of graph that are larger than threshold 0.5 are selected as final character boundaries.

TABLE 7. Results of Switcher-Segmenter model in Table 6. We denote testset language as Table 3.

	Chi	Eng	Kor	Chi+Eng	Chi+Kor	Eng+Kor	Chi+Eng+Kor	Average
Proposed method	94.74	77.01	97.07	87.23	97.10	87.46	90.87	90.21
Switcher-Segmenter	61.38	37.67	82.34	28.35	35.53	31.20	23.08	42.79

summarized in TABLE 5, it shows similar performance to the projection profile method.

Then, we apply the REINFORCE algorithm and evaluate the performance. In this case, we conduct experiments with $\eta = 0.1$ and 0.4 as well as the η-scheduling scheme, which was discussed in Sec. III-D. As shown in the table, the reinforcement learning largely improves the accuracy of the segmenter. Also, η-scheduling scheme improves the overall performance, showing that this scheme helps the convergence of the segmenter. Examples of boundary predictions with/without the reinforcement learning are shown in Figure 9.

C. ADDITIONAL EXPERIMENTS

In order to improve the efficiency and the performance of the proposed method, we have also conducted additional experiments changing architectures and normalization methods.

1) INTEGRATING SEGMENTER AND SWITCHER

While using multiple independent recognizers, we have tried to integrate the segmenter and switcher in Figure 1, so that we can predict boundaries and language classes at the same time. TABLE 6 shows an example of our Switcher-Segmenter (SS) architectures, which is basically the concatenation of the segmenter and the switcher presented in TABLE 2: Our SS network is designed to yield boundary predictions in the middle of the network and language predictions at the end of the network. To learn the features for both boundary prediction and language prediction, we increase the channels in convolution layers and the units in LSTM layers of the segmenter part. Also, to use sequential inputs from LSTM layers, we reduce the kernel height size of the convolution layers in the switcher part to 1 and decrease the number of channels for complexity reduction.

For the training of this two-head network, we need to perform sequence-to-sequence learning in addition to our reinforcement learning, where one sequence is the output of the right-hand-side head of our SS network whose length is the width of an input image and the other is the ground truth

TABLE 8. Architectures of unified Recognizer (R_{uni}).

R_{uni}					
Conv(42, 5×5, 1×1)					
AvgPool(2×2, 2×2)					
Conv(112, 5×5, 1×1)					
BatchNorm					
AvgPool(2×2, 2×2)					
Conv(840, 5×8, 1×1)					
BatchNorm					
FC(84) FC(53) Softmax	FC(84) FC(47) Softmax	FC(168) FC(1818) Softmax	FC(84) FC(20) Softmax	FC(84) FC(22) Softmax	FC(84) FC(29) Softmax
English Output	Special Output	Chinese Output	Korean Output		

TABLE 9. Results of unified recognizer models in Table 8. We denote testset language as Table 3.

	Chi	Eng	Kor	Chi+Eng	Chi+Kor	Eng+Kor	Chi+Eng+Kor	Average
Proposed method	94.74	77.01	97.07	87.23	97.10	87.46	90.87	90.21
Unified Recognizer	93.74	75.15	95.28	86.00	96.32	87.17	89.66	89.05

TABLE 10. Results of Recognizer and Switcher with other normalization methods. "LayerNorm" means the model with the layer normalization [36]. "SELU" means the models with SELU activations [37]. We denote testset words language as Table 3.

	Chi	Eng	Kor	Chi+Eng	Chi+Kor	Eng+Kor	Chi+Eng+Kor	Average
Proposed method	94.74	77.01	97.07	87.23	97.10	87.46	90.87	90.21
Recognizer (LayerNorm)	94.83	73.48	96.75	85.93	96.74	87.46	90.01	89.32
Recognizer (SELU)	94.20	73.48	96.34	84.97	96.82	87.37	89.41	88.94
Switcher (LayerNorm)	94.83	78.67	90.89	87.51	93.35	85.91	90.27	88.77
Switcher (SELU)	95.10	78.18	89.83	87.71	91.65	85.42	87.86	87.96

label whose length is the number of characters in the word. In order to address this problem, we use a CTC loss that enables the sequence-to-sequence learning [1]. Similarly, in the inference phase, the SS network yields W inference results, for a $H \times W$ input, while we only need one language class for each segmented character image. To address this problem, we first extract character boundaries with the outputs of the left-hand-side head in TABLE 6, and consider the averaged label on a corresponding interval as a language label for that sub-image.

The results of this SS model are in TABLE 7. Although we have tried the training for a range of configurations, SS model shows poor performance compared to the proposed method. We believe that this comes from the failure of reward evaluations. In order to train a model using the reinforcement learning framework, rewards should reflect the qualities of actions faithfully. However, our integrated model predicts language classes as well as boundary labels, and promising actions drawn from the policy distribution often get bad rewards due to incorrect language classes.

2) INTEGRATING RECOGNIZERS

We have also tried to develop a unified recognizer that shares convolution layers while using the same segmenter and switcher model. As shown in TABLE 8, the network has six heads to predict character classes. Since this unified recognizer yields character classes for all languages, it is trained to yield the NR class for foreign characters. For the fair comparison, we set the number of channels in shared convolution layers to the total number of channels in recognizers and use the same architectures of fully connected layers in TABLE 1. In the test phase, we select outputs of this unified recognizer based on the result of the switcher. As shown in TABLE 9, this unified recognizer shows slightly lower performance even with the increased inference time. Therefore, we believe that individual recognizers can exploit the structural properties of each language more effectively.

3) NORMALIZATION METHODS ON MODELS

To improve the performance, we have trained our recognizers by employing other normalization methods such as layer normalization [36] and SELU [37] on convolution layers. However, as shown in TABLE 10, the results are similar to those of the batch normalization. We believe that the training of CNNs for the single character recognition is not a very challenging problem and the conventional batch normalization shows good performance.

VI. CONCLUSION

In this article, we have proposed a multilingual OCR system integrating three neural blocks (a segmenter, a switcher, and multiple recognizers for different languages), and the reinforcement learning of the segmenter. Unlike conventional methods for multi-language OCR systems, we aimed to optimize the edit distance of recognition results (i.e., the overall performance of OCR systems) and achieved the goal with the reinforcement learning method. Experimental results show that our method outperforms conventional methods. We will release our code and datasets for further research and development.

REFERENCES

[1] B. Shi, X. Bai, and C. Yao, "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.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017.

[2] D. Kumar Sahu and M. Sukhwani, "Sequence to sequence learning for optical character recognition," 2015, *arXiv:1511.04176*. [Online]. Available: http://arxiv.org/abs/1511.04176

[3] R. Smith. (2016). *Tesseract Blends Old and New OCR Technology*. [Online]. Available: https://github.com/tesseract-ocr/docs/tree/master/das_tutorial2016

[4] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation methods for character recognition: From segmentation to document structure analysis," *Proc. IEEE*, vol. 80, no. 7, pp. 1079–1092, Jul. 1992.

[5] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose, "A novel adaptive morphological approach for degraded character image segmentation," *Pattern Recognit.*, vol. 38, no. 11, pp. 1961–1975, Nov. 2005.

[6] X. Jia, X. Wang, W. Li, and H. Wang, "A novel algorithm for character segmentation of degraded license plate based on prior knowledge," in *Proc. IEEE Int. Conf. Autom. Logistics*, Aug. 2007, pp. 249–253.
[7] P. Sahare and S. B. Dhok, "Multilingual character segmentation and recognition schemes for Indian document images," *IEEE Access*, vol. 6, pp. 10603–10617, 2018.
[8] B. Q. Long Mai, T. H. Huynh, and A. D. Doan, "An independent character recognizer for distantly acquired mobile phone text images," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2016, pp. 85–90.
[9] Y. S. Chernyshova, A. V. Sheshkus, and V. V. Arlazarov, "Two-step CNN framework for text line recognition in camera-captured images," *IEEE Access*, vol. 8, pp. 32587–32600, 2020.
[10] H. Zheng, J. Wang, Z. Huang, Y. Yang, and R. Pan, "Chinese/English mixed character segmentation as semantic segmentation," 2016, *arXiv:1611.01982*. [Online]. Available: http://arxiv.org/abs/1611.01982
[11] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys.-Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.
[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.
[13] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What is wrong with scene text recognition model comparisons? Dataset and model analysis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4715–4723.
[14] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "PhotoOCR: Reading text in uncontrolled conditions," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 785–792.
[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, Granada, Spain, 2011, p. 5.
[16] M. Liao, J. Zhang, Z. Wan, F. Xie, J. Liang, P. Lyu, C. Yao, and X. Bai, "Scene text recognition from two-dimensional perspective," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8714–8721.
[17] F. Yin, Y.-C. Wu, X.-Y. Zhang, and C.-L. Liu, "Scene text recognition with sliding convolutional character models," 2017, *arXiv:1709.01727*. [Online]. Available: http://arxiv.org/abs/1709.01727
[18] Z. Chen and X. Ding, "Rejection algorithm for mis-segmented characters in multilingual document recognition," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, 2003, pp. 746–749.
[19] T. M. Breuel, "High performance text recognition using a hybrid convolutional-LSTM implementation," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Nov. 2017, pp. 11–16.
[20] F. Asad, A. Ul-Hasan, F. Shafait, and A. Dengel, "High performance OCR for camera-captured blurred documents with LSTM networks," in *Proc. 12th IAPR Workshop Document Anal. Syst. (DAS)*, Apr. 2016, pp. 7–12.
[21] Y. Xia, B.-H. Xiao, C.-H. Wang, and R.-W. Dai, "Integrated segmentation and recognition of mixed Chinese/English document," in *Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 2, Sep. 2007, pp. 704–708.
[22] J. Tian, R. Wang, G. Wang, J. Liu, and Y. Xia, "A two-stage character segmentation method for chinese license plate," *Comput. Electr. Eng.*, vol. 46, pp. 539–553, Aug. 2015.
[23] V. Bansal and R. M. K. Sinha, "Segmentation of touching and fused devanagari characters," *Pattern Recognit.*, vol. 35, no. 4, pp. 875–893, Apr. 2002.
[24] L. Zheng, A. H. Hassin, and X. Tang, "A new algorithm for machine printed arabic character segmentation," *Pattern Recognit. Lett.*, vol. 25, no. 15, pp. 1723–1729, Nov. 2004.
[25] Y. Mei, X. Wang, and J. Wang, "A Chinese character segmentation algorithm for complicated printed documents," *Int. J. Signal Process., Image Process. Pattern Recognit.*, vol. 6, no. 3, pp. 91–100, 2013.
[26] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 7, pp. 690–706, Jul. 1996.
[27] G. Nagy, "Twenty years of document image analysis in PAMI," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 38–62, Jan. 2000.
[28] A. Kumar, M. Yadav, T. Patnaik, and B. Kumar, "A survey on touching character segmentation," *Int. J. Eng. Adv. Technol.*, vol. 2, no. 3, pp. 2249–8958, Feb. 2013.
[29] Y. Rao, D. Lin, J. Lu, and J. Zhou, "Learning globally optimized object detector via policy gradient," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6190–6198.
[30] J. Ryu, H. I. Koo, and N. I. Cho, "Word segmentation method for handwritten documents based on structured learning," *IEEE Signal Process. Lett.*, vol. 22, no. 8, pp. 1161–1165, Aug. 2015.
[31] H. I. Koo and N. I. Cho, "Robust skew estimation using straight lines in document images," *J. Electron. Imag.*, vol. 25, no. 3, Jun. 2016, Art. no. 033014.
[32] H. I. Koo and N. I. Cho, "Text-line extraction in handwritten Chinese documents based on an energy minimization framework," *IEEE Trans. Image Process.*, vol. 21, no. 3, pp. 1169–1175, Mar. 2012.
[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[36] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*. [Online]. Available: http://arxiv.org/abs/1607.06450
[37] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 971–980.

JAEWOO PARK (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include image processing, computer vision, and machine learning.

EUNJI LEE (Graduate Student Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2018, where she is currently pursuing the Ph.D. degree in electrical and computer engineering. Her research interests include image processing, computer vision, and machine learning.

YOONSIK KIM received the B.S. degree in electronics engineering from Chung-Ang University, Seoul, South Korea, in 2014. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University, Seoul. His research interests include image processing, computer vision, and machine learning.

ISAAC KANG (Graduate Student Member, IEEE) received the B.S. degree from the College of Liberal Studies, Seoul National University, Seoul, South Korea, in 2019. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University. His research interests include image processing, computer vision, and machine learning.

HYUNG IL KOO (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Department of Electrical Engineering and Computer Science, Seoul National University, Seoul, South Korea, in 2002, 2004, and 2010, respectively. From 2010 to 2012, he was a Research Engineer with Qualcomm Research Korea. He joined the Department of Electrical and Computer Engineering, Ajou University, in 2012, where he is currently an Associate Professor. His research interests include computer vision and machine learning.

NAM IK CHO (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1986, 1988, and 1992, respectively. From 1991 to 1993, he was a Research Associate with the Engineering Research Center for Advanced Control and Instrumentation, Seoul National University. From 1994 to 1998, he was an Assistant Professor of electrical engineering with the University of Seoul. In 1999, he joined the Department of Electrical and Computer Engineering, Seoul National University, where he is currently a Professor. His research interests include image processing, adaptive filtering, digital filter design, and computer vision.