EASR

Engineering and Applied Science Research

https://www.tci-thaijo.org/index.php/easr/index

Published by the Faculty of Engineering, Khon Kaen University, Thailand

CycleAugment: Efficient data augmentation strategy for handwritten text recognition in historical document images

Sarayut Gonwirat and Olarik Surinta*

Multi-agent Intelligent Simulation Laboratory (MISL), Department of Information Technology, Faculty of Informatics, Mahasarakham University, Mahasarakham 44150, Thailand

Received 24 November 2021
Revised 4 February 2022
Accepted 25 February 2022

Abstract

Predicting the sequence pattern of the handwritten text images is a challenging problem due to various writing styles, insufficient training data, and also background noise appearing in the text images. The architecture of the combination between convolutional neural network (CNN) and recurrent neural network (RNN), called CRNN architecture, is the most successful sequence learning method for handwritten text recognition systems. For handwritten text recognition in historical Thai document images, we first trained nine different CRNN architectures with both training from scratch and transfer learning techniques to find out the most powerful technique. We discovered that the transfer learning technique does not significantly outperform scratch learning. Second, we examined training the CRNN model by applying the basic transformation data augmentation techniques: shifting, rotation, and shearing. Indeed, the data augmentation techniques provided more accurate performance than without applying data augmentation techniques. However, it did not show significant results. The original training strategy aimed to find the global minima value and not always solve the overfitting problems. Third, we proposed a cyclical data augmentation strategy, called CycleAugment, to discover many local minima values and prevent overfitting. In each cycle, it rapidly decreased the training loss to reach the local minima. The CycleAugment strategy allowed the CRNN model to learn the input images with and without applying data augmentation techniques to learn from many input patterns. Hence, the CycleAugment strategy consistently achieved the best performance when compared with other strategies. Finally, we prevented image distortion by applying a simple technique to the short word images and achieved better performance on the historical Thai document image dataset.

Keywords: Convolutional recurrent neural network, Handwritten text recognition, Data augmentation, Deep learning, Training strategy

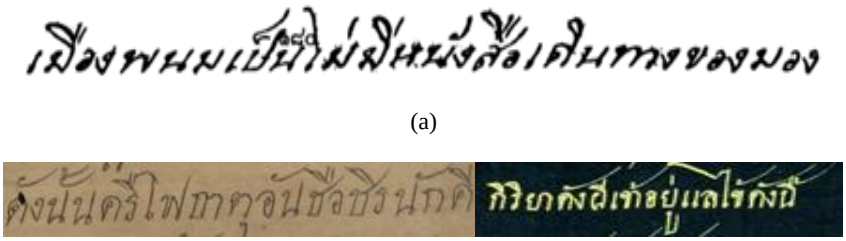
1. Introduction

The offline text recognition system is a vision-based application that automates extracting information from handwritten and printed manuscripts and transforms images into digitally readable text that is editable and comfortable to store and retrieve. Earlier research focused on character recognition which recognized isolated characters [1-4]. However, a few studies concentrated on the recognition of handwritten text. This is because it takes more effort to segment handwritten text into individual characters [5-7]. Due to messy handwriting, various writing styles, and cursive texts, as shown in Figure 1, it is difficult to solve by segmenting characters and then recognizing them by traditional optical character recognition (OCR). Character sequence learning is more suitable for word recognition [8, 9]. Hence, an effective feature-based sliding window and sequence learning methods are applied to recognize each character and then transcript to words [10-12]. However, handwritten text recognition (HTR) methods mainly focus on word recognition and have become a more prominent research domain nowadays.

Deep learning methods have become the principal method in various computer vision applications, such as object detection, object recognition, speech recognition, and natural language processing. Further, convolutional neural network (CNN) architectures, one of the deep learning methods, are widely proposed for feature extraction and image classification. CNN is also proposed to address the challenge of word recognition [13-15]. In addition, CNN and recurrent neural networks (RNNs), which are the famous sequence learner architectures, were proposed to recognize both printed and handwritten words [14, 15] and achieved a high accuracy performance. Consequently, state-of-the-art in handwritten character recognition is a combination of CNN and RNN, called convolutional recurrent network (CRNN). The CRNN also proposed solving problems in many text recognition fields such as scene text and video subtitle recognition.

Moreover, handwritten text recognition has been applied in many languages, such as English, Chinese, Arabic, Indian, and Amharic [13, 14, 16-19]. Particularly, historical manuscripts contain cursive writing, noisy background, and differing word spelling from an ancient and insufficient lexicon for transcription. The challenge of Thai handwritten character recognition is that the Thai language does not have an exact rule to split the sentences and no space between words. For explicit prediction, it is demanding to segment sentences into tokenized words.

*Corresponding author. Tel.: +66 4375 4359
Email address: olarik.s@msu.ac.th
doi: 10.14456/easr.2022.50



(a) (b) (c)

Figure 1 Examples of historical Thai handwritten texts from (a) Thai archive, (b) Phra Narai Medicine, and (c) King Rama V, volume 1, medicine manuscripts

The main contribution of this paper is to present the new data augmentation strategy, namely CycleAugment. The proposed data augmentation strategy mainly focuses on minimizing the validation loss and avoiding overfitting. We achieve our goal with a simplistic strategy and implementation. Our research is motivated by Huang et al. [20], who proposed the cyclic cosine annealing method that calculated the learning rate in every epoch and then started the new learning rate at the beginning of a new cycle.

Furthermore, training the CRNN model usually allows choosing only to train the CRNN model with or without applying data augmentation techniques. We offer the CycleAugment strategy that provides the ability to train the CRNN model with and without applying data augmentation techniques simultaneously. Importantly, our CycleAugment strategy confirms that it can handle every CRNN architecture.

We evaluate the efficiency of the CycleAugment strategy on several CRNN architectures for handwritten word recognition on Thai archive manuscripts. To show the importance of the CycleAugment strategy, we compared it to the original data augmentation strategy. The results showed that the CycleAugment strategy significantly decreased the character error rate (CER). The CycleAugment strategy achieved the CER value of 5.43 and the original data augmentation strategy obtained the CER value of 7.31 on the Thai archive manuscript.

The remainder of this paper is organized as follows. The related work is briefly described in Section 2. Section 3 deeply explains the proposed CRNN architecture and proposed CycleAugment strategy. Section 4, present the Thai historical document dataset, training strategy, and experimental evaluation. The discussion is presented in Section 5. Finally, the last section gives the conclusion and future direction

2. Related work

In this section, we survey the HTR task based on deep learning techniques. We also study the transfer learning and data augmentation techniques that improve the performance of deep learning.

2.1 Handwritten text recognition

Text recognition systems have been proposed for several applications, such as scene text recognition [21-24], video subtitle recognition [17, 25], and handwritten text recognition in many languages [14, 16, 19]. Currently, most of the proposed HTR methods are based on the CNNs and RNNs architectures.

For HTR, Abdurahman et al. [16] proposed a convolutional recurrent neural network architecture, called AHWR-Net, to recognize Amharic words. The AHWR-Net architecture was divided into feature extraction, sequence modeling, and classification. First, they created a CNN model and compared their proposed CNN model with state-of-the-art CNN models: DenseNet-121, ResNet-50, and VGG-19. These CNN models were also proposed to extract the feature from the Amharic word images. Second, the RNN architecture was proposed as the sequence model to train spatial features extracted from the previous step. Finally, the probability distributions, which was the output of the RNN method, were classified using a connectionist temporal classification algorithm (CTC). In addition, Butt et al. [19] built a robust Arabic text recognition system using the CNN-RNN attention model from natural scene images. Their Arabic text recognition system addressed the challenge of working with texts in different sizes, fonts, colors, orientation, and brightness.

Furthermore, Ameryan & Schomaker [14] proposed a high-performance word classification using homogeneous CNN and long short-term memory (LSTM) networks. First, for the CNN model, they created five CNN layers. Each CNN layer contained a convolutional layer, normalization method, nonlinear rectified linear unit (ReLU), and max-pooling layer. Second, for the LSTM, bidirectional-LSTMs with three layers were used. Third, the CTC decoding was attached as the output of their network. Finally, invented ensemble system was proposed, which included five networks. The outputs of each network were sent to vote using the plurality vote method.

2.2 Thai handwritten text recognition

There is a small amount of research that focuses on Thai handwritten text recognition. In 2019, Chamchong et al. [26] proposed hybrid deep neural networks that combined three convolutional layers and two bidirectional gated recurrent unit (BiGRU) layers, namely 3CNN+BiGRU. The 3CNN+BiGRU was followed by softmax and CTC loss functions. The computational time was compared for both bidirectional LSTM (BiLSTM) and BiGRU revealing that BiGRU is faster than BiLSTM. Therefore, the 3CNN+BiGRU showed the best character error rate (CER) of 12.1% on the Thai archive dataset when time step and RNN size were set as 32 and 128.

In 2020, Srinilta & Chatpoch [27] proposed a deep learning method for multi-task learning on three scripts: Thai, Devanagari, and Latin. The deep learning method was divided into three main layers: CNN, RNN, and CTC. First, the CNN layer was trained based on ResNet50 architecture for multi-task learning, followed by the BiGRU layer. Second, the output of the BiGRU layer in the first step was trained using different BiGRU layers and then followed by the CTC layer. For example, the BiGRU-Thai layer was aimed to train and recognize only Thai scripts.

In 2021, Chamchong et al. [28] created four CNN layers: convolutional, ReLU activation, max-pooling, and dropout layers. Then, two BiGRU and dropout layers were added to the last CNN layer. In their method, the dropout layers were set as 0.2. Furthermore, to decrease the training loss value, they compared results based on two optimization algorithms: SGD and RMSprop. The experimental results showed that the RMSprop optimization outperformed the SGD optimization algorithm on the standard Thai handwritten dataset.

2.3 Improve the deep learning performance with transfer learning and data augmentation techniques

In deep learning, achieving high performance is generally accompanied by various convolutional layers and training images [29]. The very deep convolutional layers and limited training samples often attend to the overfitting problem [30]. The deep convolutional layers directly affect the deep learning model that makes it hard to generalize new samples. Transfer learning, data augmentation, dropout, and reducing the complexity of the deep learning architecture are suggested to address the overfitting problem [31-34].

Gonwirat & Surinta [31] trained CNN models (including Inception-ResNetV2 and VGG19 architectures) based on two training methods, scratch learning and transfer learning. The comparison results showed that the transfer achieved high accuracy when evaluated using 5-fold cross-validation (5-cv) and 10-cv methods. As a result, the VGG19 architecture, which included 19 layers and was designed as a stacked network, outperformed Inception-ResNetV2 when training with transfer learning on the THI-C68 dataset. It also improved the recognition speed.

Pawara et al. [33] applied six data augmentation techniques: rotation, blur, scaling, contrast, illumination, and projective to the original image. In their method, first, the training examples increased 10 to 25 times larger than the original data. Second, they trained the CNN models with original and augmentation images, called offline training strategy. Moreover, Enkvetchakul & Surinta [34] trained the CNN models using three training strategies: offline, online, and mixed training. Six data augmentation techniques were applied with an online training strategy while training the CNN model: width and height shift, rotation, zoom, brightness, cutout, and mixup. The online training strategy was much faster than training with offline strategy. This did not increase the number of training examples, however, it transformed the original image using augmentation techniques while training. Hence, the CNN models could learn from the new images in each epoch.

3. The convolutional recurrent neural network

In this section, we present the convolutional recurrent neural network (CRNN) framework for Thai handwritten text recognition of historical document images with a new data augmentation strategy. Firstly, convolutional neural networks (CNNs) are described. Secondly, two recurrent neural networks (RNNs) (long short-term memory and gated recurrent unit) are briefly detailed. Thirdly, detail of the connectionist temporal classification (CTC) decoding is presented for the evaluation metric. Finally, the proposed cyclical data augmentation strategy, namely CycleAugment, is presented. The proposed framework is explained as follows.

3.1 Overview of the CRNN architecture

The CRNN network is illustrated in Figure 2. The CRNN has only one input. Our framework also supports both images of a group of words and short words as for the input. In the CNN architecture, we propose eight different CNN architectures to find the best base CNN model. For the RNN network, we propose two layers of bidirectional RNN networks and connect them to the CNN architecture. Hence, the outputs of the bidirectional RNN network are then classified using the softmax function. The output of the CRNN is a matrix containing character probabilities for each time step. Further, the CTC decoding is attached at the last layer to decode the probability of characters to make the final text output. Our framework can predict a maximum of 94 members in total, including characters, numbers, and blanks (space). The configurations of all CRNN architectures are shown in Table 1.

Furthermore, we propose the cyclical data augmentation strategy (CycleAugment). The CycleAugment strategy provides the CRNN model to train handwritten text images concurrently with and without applying data augmentation techniques. CycleAugment is a powerful strategy for obtaining various local optimal loss values in each cycle until they reach a minimum value at the end of training.

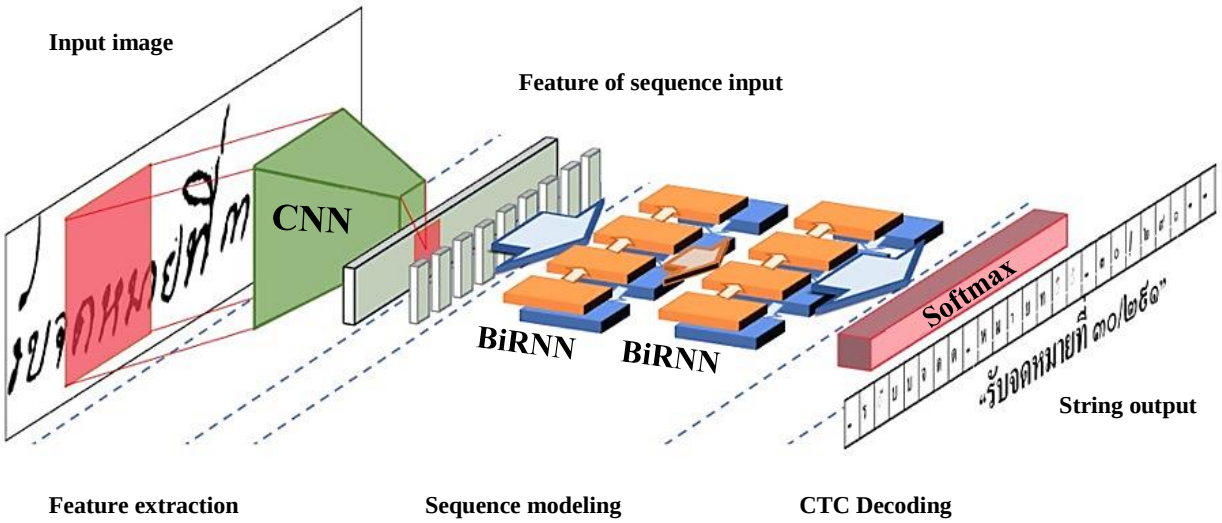


Figure 2 Overview framework of convolutional recurrent neural networks

Table1 Configuration details of CRNN architectures

CCNet	**mCCNet-64**	**mCCNet-512**	**mVGG16**	**mVGG19**	**mResNet50**	**mDenseNet-121**	**mMobileNet-V2**	**mEfficientNet-B1**
6 weighted layers	7 weighted layers	7 weighted layers	14 weighted layers	16 weighted layers	26 weighted layers	43 weighted layers	23 weighted layers	29 weighted layers
Input image (64, 504, 1)			Input image (64, 504, 3)					
Conv3-16 Maxpool2-s2	Conv3-16 Maxpool2-s2	Conv3-16 Maxpool2-s2	Conv3-16 Conv3-16 Maxpool2-s2	Conv3-64 Conv3-64 Maxpool2-s2	Conv7-64 Maxpool3-s2	Conv7-64 Maxpool3-s2	Conv3-32-s2 DwConv3-32	Conv3-32-s2 DwConv3-32
Conv3-32 Maxpool2-s2	Conv3-32 Maxpool2-s2	Conv3-32 Maxpool2-s2	Conv3-128 Conv3-128 Maxpool2-s2	Conv3-128 Conv3-128 Maxpool2-s2	$\begin{bmatrix}\text{Conv1}-64\\ \text{Conv3}-64\\ \text{Conv1}-256\end{bmatrix}$ x3	$\begin{bmatrix}\text{Conv1}-128\\ \text{Conv3}-32\end{bmatrix}$ x6 Conv1-128 Avgpool2-s2	Conv1-16 Conv1-96 DwConv3-96	Conv1-16 Conv1-96 DwConv3-96 SE
Conv3-32 Maxpool2-s2	Conv3-32 Maxpool2-s2	Conv3-32 Maxpool2-s2	Conv3-256 Conv3-256 Conv3-256 Maxpool2-s2	Conv3-256 Conv3-256 Conv3-256 Conv3-256 Maxpool2-s2	$\begin{bmatrix}\text{Conv1}-128\\ \text{Conv3}-128\\ \text{Conv1}-512\end{bmatrix}$ x4	$\begin{bmatrix}\text{Conv1}-128\\ \text{Conv3}-32\end{bmatrix}$ x12 Conv1-512	$\begin{bmatrix}\text{Conv1}-24\\ \text{Conv1}-144\\ \text{DwConv3}-144\end{bmatrix}$ x2	$\begin{bmatrix}\text{Conv1}-24\\ \text{Conv1}-144\\ \text{DwConv3}-144\\ \text{SE}\end{bmatrix}$ x2
-	-	-	Conv3-512 Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv3-512 Conv3-512	-	-	$\begin{bmatrix}\text{Conv1}-32\\ \text{Conv1}-192\\ \text{DwConv3}-192\end{bmatrix}$ x3	$\begin{bmatrix}\text{Conv1}-40\\ \text{Conv1}-240\\ \text{DwConv5}-240\\ \text{SE}\end{bmatrix}$ x2
-	Conv1-64	Conv1-512	Conv1x1-512					
Global average pooling								
Bidirectional RNN-(N)								
Bidirectional RNN-(N)								
FC, Softmax (94)								
CTC decoding								

3.2 Convolutional neural network

In recent years, many CNN architectures have been proposed to enhance the performance of image classification. This section describes the configuration of the different CNN architectures evaluated in this paper to solve handwritten text recognition. First we explain the CNN architecture proposed by Chamchong et al. [26], especially for handwritten text recognition, called CCNet. Second, we describe modification of CCNet (mCCNet) by adding a 1x1 convolutional layer (Conv1) that mainly reduced the parameters of model CC. Finally, we describe modification of state-of-the-art CNN architectures, including VGG16 and VGG19, ResNet50, DenseNet121, MobileNetV2, and EfficientNetB1.

3.2.1 CCNet

Chamchong et al. [26] proposed a simple CNN that included three blocks of convolutional and max-pooling layers. Each block contained a convolutional layer with 3x3 kernel sizes (Conv3), a max-pooling layer using 2x2 kernel sizes (Maxpool2), and a stride of 2 (s2), in order. The convolutional layers in the first, second, and third blocks were 16, 32, and 32 feature maps. Also, the ReLU activation function and batch normalization (BN) were added to the last block.

3.2.2 Modified CCNets

We modified CCNet (mCCNet) by attaching Conv1x1 to reduce the CNN parameters and introduce new nonlinearity into the network. For the mCCNet-64 and mCCNet-512 models, we implemented the Conv1 layer with feature map sizes of 64 and 512, respectively.

3.2.3 Modified VGGs

The modified VGG16 (mVGG16) and VGG19 (mVGG19) were built based on the VGG16 and VGG19 [35], respectively. These networks comprised a convolutional layer with 3x3 kernel sizes and followed by a max-pooling layer using 2x2 kernel sizes. However, the mVGG16 and mVGG19 were cut at the end of the fourth block and we also replaced them with Conv1 of 512 feature maps.

3.2.4 Modified ResNet50

He et al. [36] proposed deep residual learning to construct a deeper network without facing the gradient vanishing problem, called ResNet. ResNet50 included five convolutional blocks in which the output of each convolutional block decreased to half size when compared to the input. For the modified ResNet50 (mResNet50), we removed convolutional blocks 4 and 5 from the network. Also, we added Conv1 with 512 feature maps at the end of convolutional block 3.

3.2.5 Modified DenseNet121

The DenseNet architecture [37] was proposed to collect knowledge from all previous layers and pass them to the next layer using the densely connected operation. It required long computational time. The DenseNet121 contained two major layers: dense block and transition layer. The main network consisted of a convolutional layer, max-pooling layer, dense block, three times transition layer and dense block, and classification layer. The output of each layer decreased by half size, the same as for ResNet. For the modified DenseNet121 (mDenseNet121), however, we removed layers from the second transition layer. Hence, we attached Conv1x1 with 512 feature maps.

3.2.6 Modified MobileNetV2

MobileNetV2 was proposed by Sandler et al. [38] to reduce weighted parameters of a lightweight network using depthwise separable convolutional (DwConv) layers and inverted residuals of the bottleneck block. The main network comprised two block types: residual block with a stride of 1 and block with a stride of 2 to reduce the dimensionality of the feature map. The activation function used in MobileNetV2 was the ReLU with the maximum value of 6 (ReLU6). The MobileNetV2 network consisted of DwConv, convolutional layers, seven bottleneck blocks with different repeated times, 1x1 convolutional layer, and global average pooling (GAP) layer. Since the output of layers is decreased by half size, for modified MobileNetV2 (mMobileNetV2), we removed the fourth bottleneck block and replaced them with Conv1 with 512 feature maps.

3.2.7 Modified EfficientNetB1

Tan & Le [39] designed EfficientNets to search hyperparameters of the CNN architectures, including width scaling, depth scaling, resolution scaling, and compound scaling. In addition, the squeeze-and-excitation (SE) optimization was attached to the bottleneck block of EfficientNet to construct an informative channel feature by summation with GAP. Correlation features are then found by reducing to small dimensions and transforming them to the original dimension. EfficientNet was created based on the MobileNetV2, but it varies with resolutions, channels, and repeated times. The modified EfficientNetB1 (mEfficientNetB1) is similar to mMobileNetV2, which removed the fourth bottleneck block and replaced Conv1 with 512 feature maps.

3.3 Recurrent neural network

Recurrent neural network (RNN) was a successful architecture that was proposed to create a robust model from sequential data, such as speech [40], video [41], and brain signals [42]. RNN was designed by combining feedback loop connections that allow the output from the previous states to be applied as inputs of the current state. The feedback loop was performed in the hidden layers. However, RNN also had the limitation that constructs the output from only the previous context. The RNN is computed according to the following Equation.

$$y^t = f\left(Vh^t + b_y\right) \tag{1}$$

$$h^t = \sigma(Wx^t + Uh^{t-1} + b_h) \tag{2}$$

where y^t is the output of the RNN, h^t is the hidden state of the recurrent cell at time step (t) which is calculated by current input (x^t) and the previous hidden state (h^{t-1}). To calculate the h^t, RNN can be able to learn by adjusting the weighted parameters, including weighted matrices $(W, U$ and $V)$ and bias $(b_h$ and $b_y)$. Additionally, the output function $f(x)$ is an activation function and the sigmoid function $\sigma(x)$ is applied for hidden states.

3.3.1 Bidirectional recurrent neural network

Bidirectional recurrent neural network (BiRNN) was proposed to understand sequence data better than the RNN architecture. It contained the backward $(h^{\leftarrow t})$ and forward $(h^{\rightarrow t})$ states connected to the same output layer that helped the network effectively increase the information context. The BiRNN architecture is shown in Figure 3 and is calculated as follows.

$$\vec{h}^t = \sigma\left(\vec{W}x^t + \vec{U}\vec{h}^{t-1} + \vec{b}_h\right) \tag{3}$$

$$h^{\leftarrow t} = \sigma\left(W^{\leftarrow}x^t + U^{\leftarrow}h^{\leftarrow t-1} + b^{\leftarrow}{}_h\right) \tag{4}$$

$$y^t = f\left(V^{\leftarrow}h^{\leftarrow t} + \vec{V}\vec{h}^t + b_y\right) \tag{5}$$

3.3.2 Long short-term memory

Long short-term memory (LSTM) was invented by Hochreiter & Schmidhuber [43]. It was proposed to address the limitation of the RNN architecture that could not learn a long sequence and solve the vanishing and exploding gradient problems. The gates were designed to increase the memory capacity of the cell, including the forget gate, input gate, and output gate, as shown in Figure 4(a). The output o^t of the LSTM is computed using Equation (6) which has three input vectors, including input state x^t, previous hidden state h^{t-1}, and adding new cell state c^t. The output of the hidden state is calculated by the element-wise product $(\odot)$ between the current output of LSTM and the hyperbolic tangent function $\phi(\mathrm{x})$ of the cell state (c^t) by Equation (7). The LSTM is computed from the following Equation.

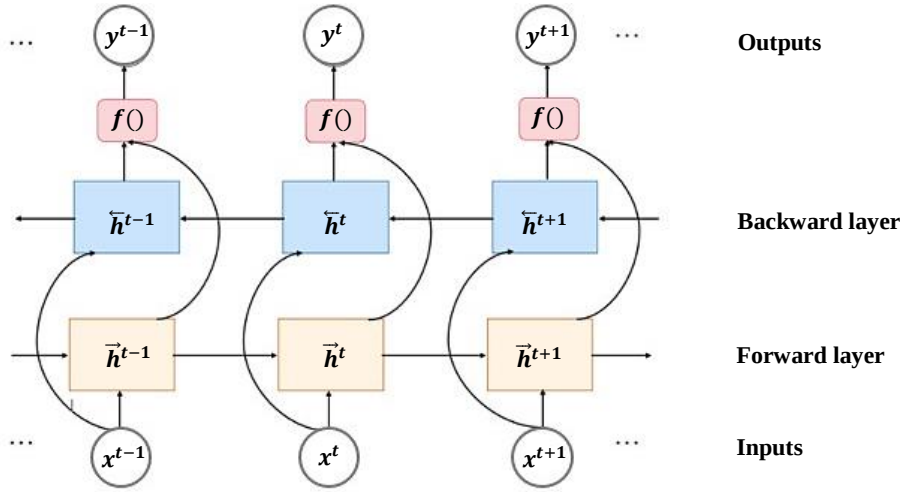
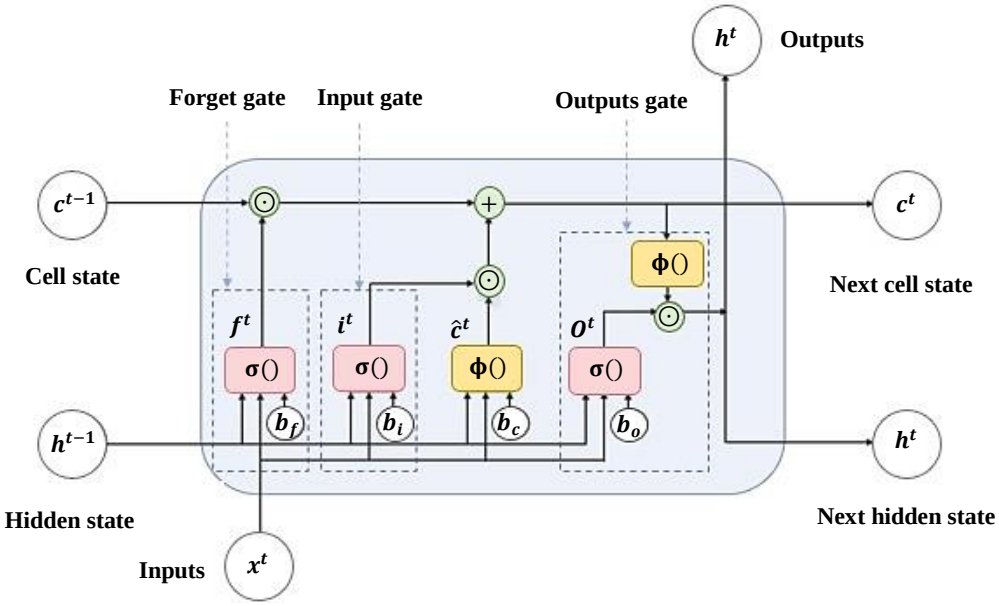
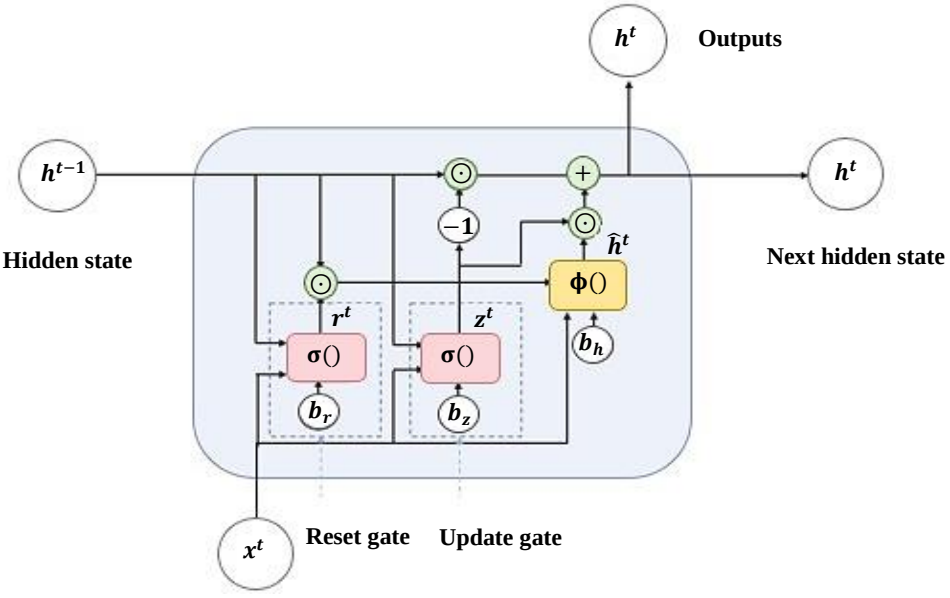


Figure 3 Illustration of bidirectional recurrent neural network



(a)



(b)

Figure 4 Illustration of the recurrent neural networks: (a) long short-term memory and (b) gated recurrent unit

$$o^t = \sigma(W_o x^t + U_o h^{t-1} + V_o c^t + b_o) \quad (6)$$

$$h^t = o^t \odot \phi(c^t) \quad (7)$$

where c^t is the cell state at time step (t) computed by Equation (8) which is designed to update the new cell state by filtering of the previous cell state (c^{t-1}) and cell candidate $\hat{c}^t$. The filtering operation is the sum of two element-wise products of the forget gate and previous cell state, and input gate and cell candidate.

$$c^t = f^t \odot c^{t-1} + i^t \odot \hat{c}^t \quad (8)$$

where forget gate (f^t), input gate (i^t) and cell candidate ($\hat{c}^t$) are computed from the following Equation.

$$f^t = \sigma\left(W_f x^t + U_f h^{t-1} + V_f c^{t-1} + b_f\right) \quad (9)$$

$$f^t = \sigma\left(W_f x^t + U_f h^{t-1} + V_f c^{t-1} + b_f\right) \quad (10)$$

$$\hat{c}^t = \phi(W_c x^t + U_c h^{t-1} + V_c c^{t-1} + b_c) \quad (11)$$

In the LSTM, $W_o, W_f, W_i, W_c, U_o, U_f, U_i, U_c, V_y, V_f, V_i$, and V_c are weight matrices and b_o, b_f, b_i, and b_c are bias parameters which are required to adjust during training.

3.3.3 Gate recurrent unit

The gate recurrent unit (GRU) was proposed by Cho et al. [44] and designed to reduce the complexity of the LSTM architecture. The GRU architecture is shown in Figure 4(b). In the GRU, the input and forget gates were replaced with a reset gate (r^t). GRU is computed from the following Equation.

$$r^t = \sigma(W_r x^t + U_r h^{t-1} + b_r) \quad (12)$$

where the hidden state h^t is computed using (13).

$$h^t = (1 - z^t) \odot h^{t-1} + z^t \odot \hat{h}^t \quad (13)$$

where the update gate (z^t) and the candidate of hidden state ($\hat{h}^t$) are computed from the following Equation.

$$z^t = \sigma(W_z x^t + U_z h^{t-1} + b_z) \quad (14)$$

$$\hat{h}^t = \phi(W_h x^t + U_h(r^t \odot h^{t-1}) + b_h) \quad (15)$$

where W_r, W_z, W_h, U_r, U_z, and U_h are weight matrices and b_r, b_z, and b_h are bias parameters which are required to adjust during training.

BiRNN with high-capacity memory was shown to be more efficient at learning a sequence of context information than was a single RNN layer. The BiRNN was also proposed to better understand sequence data that link the content from the backward state and link to the forward state.

In our proposed CRNN networks, the two bidirectional RNN layers were stacked on state-of-the-art CNN architectures. Two specials bidirectional RNNs, including BiLSTM and BiGRU, with diverse hidden unit sizes, were investigated.

3.4 Connectionist temporal classification

Connectionist temporal classification (CTC) is a conditional probability proposed to support RNN architecture that tackles various sequence problems [45], such as speed recognition and handwritten text recognition. The output of the CTC algorithm is the sequence probabilities that are decoded from the RNN output at each time step. The condition probability $p(l|x)$ is calculated by the sum of probabilities of all possible paths, as described in Equation (16).

$$p(l|x) = \sum_{\pi \in G^{-1}(l)} p(\pi|x) \quad (16)$$

where $p(\pi|x)$ is probability of possible path (π), when x is sequence input that is predicted to sequence label (l), $x = x_1, x_2, \ldots, x_T$, T is the length of the sequence, is member of L. In the handwritten text recognition problem, L is defined as 94 alphabets as shown in Table 2, and G is mapping function for transforming to l. For example, $G(\text{“}--\text{H H}-\text{e l l}-\text{l}-\text{o o}--\text{”}) =$ “Hello” or $G(\text{“}--\text{ส ส}-\text{วั ส ส}-\text{ด ดี}-\text{”}) =$ “สวัสดี” (in Thai). The $p(\pi|x)$ is computed as in the following equation.

$$p(\pi|x) = \prod_{t=1}^{T} x_\pi^t \quad (17)$$

where x_{π}^{t} is probability of having label π_t of time (t). CTC loss function is applied for training set (D) of pair input image and sequence label $(x_{i,}, l_i) \in D$. The CTC loss function is defined as in the following equation.

$$CTC\ loss = -\sum_{(x_{i,},l_i)\in D} log\left(p(l_i|x_i)\right) \tag{18}$$

3.5 The proposed cyclical data augmentation strategy

In this section, we propose a cyclical learning method, a novel data augmentation strategy called CycleAugment, that cooperates between training the CRNN model with applying data augmentation technique and without applying data augmentation technique. The transformation data augmentation technique and the CycleAugment strategy are described as follows.

3.5.1 Transformation data augmentation technique

We applied the basic transformation data augmentation techniques that include random shifting, rotation, and shearing, called $DA(w, h, r, s)$, where w and h are parameters of the maximum random percent of width and height shifting, respectively, r is an orientation rotation in the range of 0 to 360 degrees, and s is orientation shearing in the range of 0 to 360 degrees. In our experiments, the default parameters of the $DA()$ were defined as w_{max} = 0.15, h_{max}=0.2, r_{max}=5, and s_{max}=5.

In addition, we computed the scaling factor (α) to the $DA()$. The scaling factor affected the image directly by increasing and decreasing the image transformation.

3.5.2 CycleAugment strategy

Data augmentation is a fundamental process that can reduce overfitting but optimizing the data augmentation parameters is necessary to effectively minimize the loss during training. Therefore, we proposed the new cyclical data augmentation strategy, called the CycleAugment, to minimize effectively the validation loss and handle the overfitting problem. It was motivated by Huang et al. [20]. In their method, the cyclic learning rate, which is the cycle of the adaptive learning rate, starts at the maximum number in each cycle and then decreases until the minimum number. Instead of adjusting the learning rate, our proposed method presents another approach to improve the performance of the data augmentation technique. It can achieve higher efficiency by taking the form of a cycle of adaptive data augmentation.

In the CycleAugment strategy, we first trained the CRNN model by applying the data augmentation techniques in the first half of the cycle. Hence, the training and validation losses became high at the beginning and decreased to the local minima value in each cycle. Second, we performed training without applying the data augmentation technique in the second half of the cycle to minimize the loss in the local space with low data variants. As with cyclic learning rate [20], a high learning rate at the start of the new cycle produced a high gradient for climbing out from the current local minima. As a result, the training and validation losses increased again and were ready to find new local minima. For our work, we focused on applying the data augmentation technique to increase the loss and the chance to escape the local minima. Finally, we repeated this step many times until the last cycle. We used the scaling factor as the linear decrement that starts from the maximum and decreases to the minimum values in each cycle. The equation and algorithm of the CycleAugment strategy are described in Equation (19) and Algorithm 1.

Algorithm 1. CycleAugment strategy for training CRNN

Input: model (m), number of max epochs (T), number of cycles (N)
$M = \lfloor \frac{T}{N} \rfloor$ is the number of epochs per cycle.
for epoch t = 1 **to** T **do**:
    if $mod(t, M) > \frac{M}{2}$: // determine the half of the cycle.
        Training model m without data augmentation
    else:
        $\alpha_t = a(t)$ // calculate a scaling factor (α) using Equation (19)
        Adjust the scaling factor (α_t) of w, h, r, and s
        Training model m with data augmentation $DA(w, h, r, s)$
end for
Output: trained model m

$$a(t) = 2 * (\alpha_{max} - \alpha_{min})(\frac{T - t}{T}) \tag{19}$$

where α is the scaling factor, t is the epoch, T is the maximum epoch.

4. Experimental results

4.1 Thai archive manuscript dataset

The handwritten text manuscripts used in our experiments were Thai archive manuscripts [26] collected from Thailand's national library. They were written in approximately 1902 AD using 94 Thai characters and contained in 140 manuscripts. The 94 characters are shown in Table 2. A sample of the Thai archive manuscripts is shown in Figure 5. In this dataset, the handwritten text images were extracted from 140 manuscripts and contained 3,446-word images.

Ground Truth: ที่ ๑๔/๔๔

Ground Truth: พระที่นั่งวิมานเมฆ

Ground Truth: วันที่ ๕

Ground Truth: เมษายน

Ground truth: รัตนโกสินทรศก๓๕๑๒๑

(a) (b)

Figure 5 Illustrated of Thai archive manuscript dataset. Examples (a) of the Thai archive manuscript and (b) word images and ground truths

Table 2 The categories of Thai characters and other symbols

Types	Number of members	Members									
Consonants	44	ก	ข	ฃ	ค	ฅ	ฆ	ง	จ	ฉ	ช
		ซ	ฌ	ญ	ฎ	ฏ	ฐ	ฑ	ฒ	ณ	ด
		ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ
		ฟ	ภ	ม	ย	ร	ล	ว	ศ	ษ	ส
		ห	ฬ	อ	ฮ						
Vowel	19	ั	ะ	า	ำ	ิ	ี	ึ	ื	ุ	ู
		เ	แ	โ	ใ	ไ	ๅ	ฤ	ฦ	็	
Tones	4	่	้	๊	๋						
Special symbols	17	ฯ	ๆ	ๅ	์	์	"	(	)	+	,
		-	.	/	x	_	–		(blank)		
Numeral	10	๐	๑	๒	๓	๔	๕	๖	๗	๘	๙
Total	**94**										

4.2 Training strategy

4.2.1 Optimization algorithms

In the deep learning algorithms, various optimization algorithms were proposed to calculate the gradients of the error function while the network back-propagation. We evaluated three optimization algorithms, that were stochastic gradient descent (SGD), Adam, and RMSprop, to converge the deep learning model and reduce the loss value while training with the same learning rate of 0.001, suitable for handwritten text recognition problem s. For other parameters, here, we used the parameters momentum = 0.9 and decay rate = 0.001 as for the SGD optimizer, discounting factor (γ) = 0.9 as for the RMSprop optimizer, and the first estimate (β_1) = 0.9, the second estimate (β_2) and epsilon (ε) = 1e-07, for the Adam optimizer.

In the experiments, we found that the Adam optimizer outperformed other optimizers. Further, all the experimental results shown in the following section were evaluated based on the Adam optimizer.

4.2.2 Transfer learning

Training the CNN model usually starts with random parameters, called scratch learning, and adjusts the weighted parameters by error gradient. Hence, the weighted parameters can extract high discriminative features from input images. Furthermore, transfer learning is derived from weighted parameters that learn from a large image dataset, called a pre-trained model. The pre-trained model contains prior knowledge of convolution filters. We can remove some top layers in the pre-trained model and attach a few new layers to the last layer of the pre-trained model.

For the transfer learning, we could train six CNN architectures: VGG16, VGG19, ResNet50, DenseNet121, MobileNetV2, and EfficientNetB1, using the pre-trained models, except only CCNet, because CCNet architecture did not provide the pre-trained model. In the RNN architectures, however, the random weighted parameters consist of Conv1x1 of 512 feature maps, global average pooling layer, BiRNN, and dense layer.

4.3 Quantitative evaluation

In this section, we evaluate CRNN architectures on the Thai archive manuscript dataset using character-level error rate (CER) as the evaluation metric. We also compare nine state-of-the-art CRNN models regarding the number of parameters and training time. Moreover, we evaluate the new data augmentation strategy (CycleAugment) and compare our CycleAugment strategy with the original data augmentation strategy. Both strategies apply data augmentation techniques based on transformation techniques, including random shifting, rotation, and shearing. In addition, we evaluate the CRNN models that train from scratch and use the transfer learning technique to understand wherewith the transfer learning technique affects the CRNN models.

The performance of the handwritten text recognition was evaluated based on the CER. CER was calculated as the minimal Levenshtein distance, which is the number of single-character modifications that change the predictive text from the ground truth transcription of the word [46]. There are three operations of the CER metric: insertion, deletion, and substitution. The CER is calculated by the following Equation:

$$CER = \frac{I + S + D}{N} \tag{20}$$

where I is the number of character insertions, S is the number of character substitutions, D is the number of character deletions, and N is the total number of characters in the target text.

4.4 Performance of different combination of CRNNs

To evaluate the performance of CRNN architectures, we resized all images to 64x496 pixels and used them as the input to the CRNN architectures. We trained all the CRNN models using the Keras framework with TensorFlow backend and trained on Google cloud with NVIDIA Tesla P100 GPU with 16GB of RAM.

For the training process, we divided the Thai archive manuscript dataset with the ratio of 70:10:20 for training, validation, and test, respectively. The nine CRNN networks (see Table 1) were combined with two types of BiRNNs: BiLSTM and BiGRU. The number of RNN sizes with 128, 256, and 512 neurons was evaluated.

The CRNN networks were trained with the following parameters: 200 epochs, batch size of 32, Adam optimizer with learning rate of 0.001, the first- and second-moment estimate values of 0.9 and 0.999, and epsilon of 1e-07.

Table 3 Comparison of the parameters and computational time between different backbones CNNs and RNN sizes

Models	No. of parameters			Training time (hh:mm)			Character error rate (%)		
	RNN sizes								
	128	256	128	128	256	128	128	256	128
CCNet-BiGRU [26]	0.49M	1.75M	6.62M	00:26	00:27	00:30	13.32	14.54	14.43
CCNet-BiLSTM [26]	0.64M	2.30M	8.78M	00:26	00:27	00:34	14.54	14.81	15.29
mCCNet-64-BiGRU	0.50M	1.75M	6.62M	00:26	00:27	00:33	15.19	16.23	16.48
mCCNet-64-BiLSTM	0.64M	2.31M	8.78M	00:26	00:27	00:30	16.09	16.64	14.36
mCCNet-512-BiGRU	0.87M	2.47M	8.03M	00:26	00:27	00:30	14.48	16.15	15.70
mCCNet-512-BiLSTM	1.13M	3.26M	10.65M	00:26	00:26	00:33	14.26	12.69	11.35
mVGG16-BiGRU	8.72M	10.32M	15.87M	00:50	00:54	01:00	11.41	11.37	14.05
mVGG16-BiLSTM	8.98M	11.10M	18.49M	00:50	00:54	01:00	9.04	12.03	14.01
mVGG19-BiGRU	11.67M	13.27M	18.82M	00:54	00:57	01:00	13.32	20.01	19.56
mVGG19-BiLSTM	11.97M	14.05M	21.44M	00:57	01:00	01:07	12.30	15.01	15.10
mResNet50-BiGRU	2.54M	4.14M	9.70M	00:37	00:40	00:43	8.40	8.22	10.77
mResNet50-BiLSTM	2.80M	4.92M	12.31M	00:37	00:40	00:47	11.16	7.29	8.21
mDenseNet121-BiGRU	2.39M	3.99M	9.55M	00:40	00:43	00:50	10.08	8.07	7.44
mDenseNet121-BiLSTM	2.65M	4.78M	12.17M	00:40	00:43	00:50	7.72	7.13	7.65
mMobileNetV2-BiGRU	0.98M	2.58M	8.14M	00:40	00:43	00:50	13.95	15.08	13.04
mMobileNetV2-BiLSTM	1.24M	3.36M	10.76M	00:40	00:43	00:50	10.91	9.13	9.73
mEfficientNetB1-BiGRU	1.06M	2.66M	8.22M	01:04	01:07	01:14	47.41	45.94	41.17
mEfficientNetB1-BiLSTM	1.32M	3.45M	10.84M	01:04	01:07	01:14	27.61	54.30	20.47

Table 3 shows a comparison of the number of parameters and computation time between different CRNN backbones. The results showed that the CCNet-BiGRU proposed by Chamchong et al. [26] provided 0.49M with the fewest parameters. It also spent less computation time of only 26 minutes. Because CCNet-BiGRU had only six weighted layers. In comparison, except mVGG16 and mVGG19 which had 14 and 16 weighted layers, other CRNN architectures had more than 20 weighted layers. When comparing the number of parameters and time spent while training the CRNN model between BiGRU and BiLSTM, we found that the BiGRU always provided fewer parameters than the BiLSTM. Therefore, the time spent while training the CRNN with BiGRU and BiLSTM was approximately the same.

In terms of the handwritten text recognition, Table 3 presents the CER value (%) in different RNN sizes (128, 256, and 512). The lowest CER value represents the best performance. In our experiments, the mDenseNet121-BiLSTM with the RNN size of 256 showed the fewest CER value of 7.13%. However, the mEfficientNetB1-BiGRU and -BiLSTM performed the worst with a CER value above 40% with BiGRU.

In the following experiments, we will continue experiments based on the best performance in each CNN architecture.

4.5 Performance of CRNN with CycleAugment strategy

In this experiment, we tested our CycleAugment strategy with all CRNN architectures to show that the proposed CycleAugment strategy obtains robust performance when training with every CRNN architecture. To discover the best CycleAugment strategy, we trained all CRNN models with 200 epochs in total and with 200 epochs, a network training of five cycles (number of epochs per cycle M =200/5). Network training using transformation data augmentation technique 20 epochs (M/2) was used and then switched to train the model without using data augmentation techniques in the following 20 epochs. Hence, it continued in the loop until the last epochs.

Table 4 presents the performance of the CycleAugment strategy. We achieved worthwhile performance when using CycleAugment with N=5. As a result, the CER value of the mEfficientNetB1-BiLSTM enormously decreased from 27.6% to only 7.74%. Consequently, the mResNet50-BiLSTM was the best CRNN model that achieved a 5.47% CER value using the CycleAugment strategy.

Table 4 Performance of different number of cycles in CycleAugment strategy

Models (RNN sizes)	Character error rate (%)					
	Number of cycles (N)					
	N=1	**N=2**	**N=3**	**N=4**	**N=5**	**N=6**
CCNet-BiGRU (128) [26]	11.40	10.12	10.89	10.14	9.46	10.26
mCCNet-64-BiLSTM (512)	10.47	13.07	13.03	12.57	13.55	12.41
mCCNet-512-BiLSTM (512)	12.97	11.17	9.74	9.21	9.66	9.14
mVGG16-BiLSTM (128)	9.50	5.70	6.20	6.59	6.29	6.03
mVGG19-BiGRU (128)	9.87	9.26	9.52	7.84	7.51	7.18
mResNet50-BiLSTM (256)	5.79	5.97	5.64	5.54	5.47	5.71
mDenseNet121-BiLSTM (256)	6.02	5.97	6.46	6.29	5.64	6.29
mMobileNetV2-BiLSTM (256)	7.75	9.35	7.95	7.73	7.64	8.36
mEfficientNetB1-BiLSTM (128)	31.46	19.83	18.52	8.17	7.74	7.30

Table 5 Performance of scratch learning different data augmentation strategies

Models (RNN sizes)	Character error rate (%)					
	No augmentation		**Data augmentation**		**CycleAugment**	
	5-cv	**Test**	**5-cv**	**Test**	**5-cv**	**Test**
CCNet-BiGRU (128) [26]	13.79 ± 0.58	13.32	13.99 ± 0.97	12.29	10.33 ± 0.67	9.46
mCCNet-64-BiLSTM (512)	18.26 ± 1.36	14.36	17.70 ± 1.71	13.78	13.34 ± 1.07	13.55
mCCNet-512-BiLSTM (512)	11.93 ± 0.76	11.35	11.55 ± 0.72	11.18	9.73 ± 0.64	9.66
mVGG16-BiLSTM (128)	8.94 ± 0.78	9.04	10.37 ± 0.74	11.71	6.77 ± 0.40	6.29
mVGG19-BiGRU (128)	13.34 ± 2.15	10.25	12.74 ± 0.56	10.10	7.62 ± 0.27	7.51
mResNet50-BiLSTM (256)	9.54 ± 1.39	7.29	7.89 ± 0.43	7.85	6.65 ± 0.40	5.47
mDenseNet121-BiLSTM (256)	7.15 ± 0.57	7.13	7.57 ± 0.60	7.44	6.55 ± 0.75	5.64
mMobileNetV2-BiLSTM (256)	10.83 ± 0.87	9.13	11.64 ± 0.88	10.47	7.93 ± 0.52	7.64
mEfficientNetB1-BiLSTM (128)	29.75 ± 2.17	27.61	29.01 ± 2.28	26.19	12.19 ± 0.78	7.74

Table 6 Performance of transfer learning different data augmentation strategies

Models (RNN sizes)	Character error rate (%)					
	No augmentation		**Data augmentation**		**CycleAugment**	
	5-cv	**Test**	**5-cv**	**Test**	**5-cv**	**Test**
mVGG16-BiLSTM (128)	9.00 ± 0.83	7.63	10.24 ± 0.60	7.31	6.88 ± 0.32	5.43
mVGG19-BiGRU (128)	13.61 ± 2.39	15.05	14.60 ± 1.45	14.89	7.64 ± 0.52	7.37
mResNet50-BiLSTM (256)	8.19 ± 0.73	7.15	7.67 ± 0.51	7.54	6.17 ± 0.53	5.69
mDenseNet121-BiLSTM (256)	7.04 ± 0.40	7.18	7.62 ± 0.40	7.48	5.82 ± 0.40	5.69
mMobileNetV2-BiLSTM (256)	10.72 ± 0.85	9.62	11.12 ± 1.17	10.18	7.88 ± 0.58	7.44
mEfficientNetB1-BiLSTM (128)	24.34 ± 3.25	27.68	24.38 ± 2.67	28.29	10.77 ± 1.12	7.60

Furthermore, to demonstrate that our CycleAugment strategy outperforms the original data augmentation strategy, we trained CRNN architecture with two different data augmentation strategies using a 5-fold cross-validation technique (5-cv). We also trained the CRNN model using the learning from scratch and transfer learning techniques. All the experimental results are shown in the following section.

We compared scratch learning and transfer learning, as shown in Table 5 and Table 6, with different training strategies, including training without applying data augmentation, training with applied transformation data augmentation techniques with scaling factor α = 1, and training with applying CycleAugment strategy with the number of cycles N= 5, α_{min} = 0.5, and α_{max}= 2.5.

As a result, the CycleAugment strategy outperformed other data augmentation strategies on both scratch learning and transfer learning. For scratch learning, the CycleAugment achieved the best CER value of 5.47% on the test set when training with mResNet50-

BiLSTM (256). For transfer learning, we found that the mVGG16-BiLSTM (128) outperformed all the CRNN architectures with the CER value of 5.43% on the test set.

Consequently, we evaluated the CRNN architectures using the 5-fold cross-validation (5-cv). We found that both scratch and transfer learning provided nearly similar CER values because we could transfer a few parameters of the CNN pre-trained models. For example, only 1 million parameters could transfer from the pre-trained ResNet50 model, while the total parameters of the mResNet50-BiLSTM (256) was 4 million.

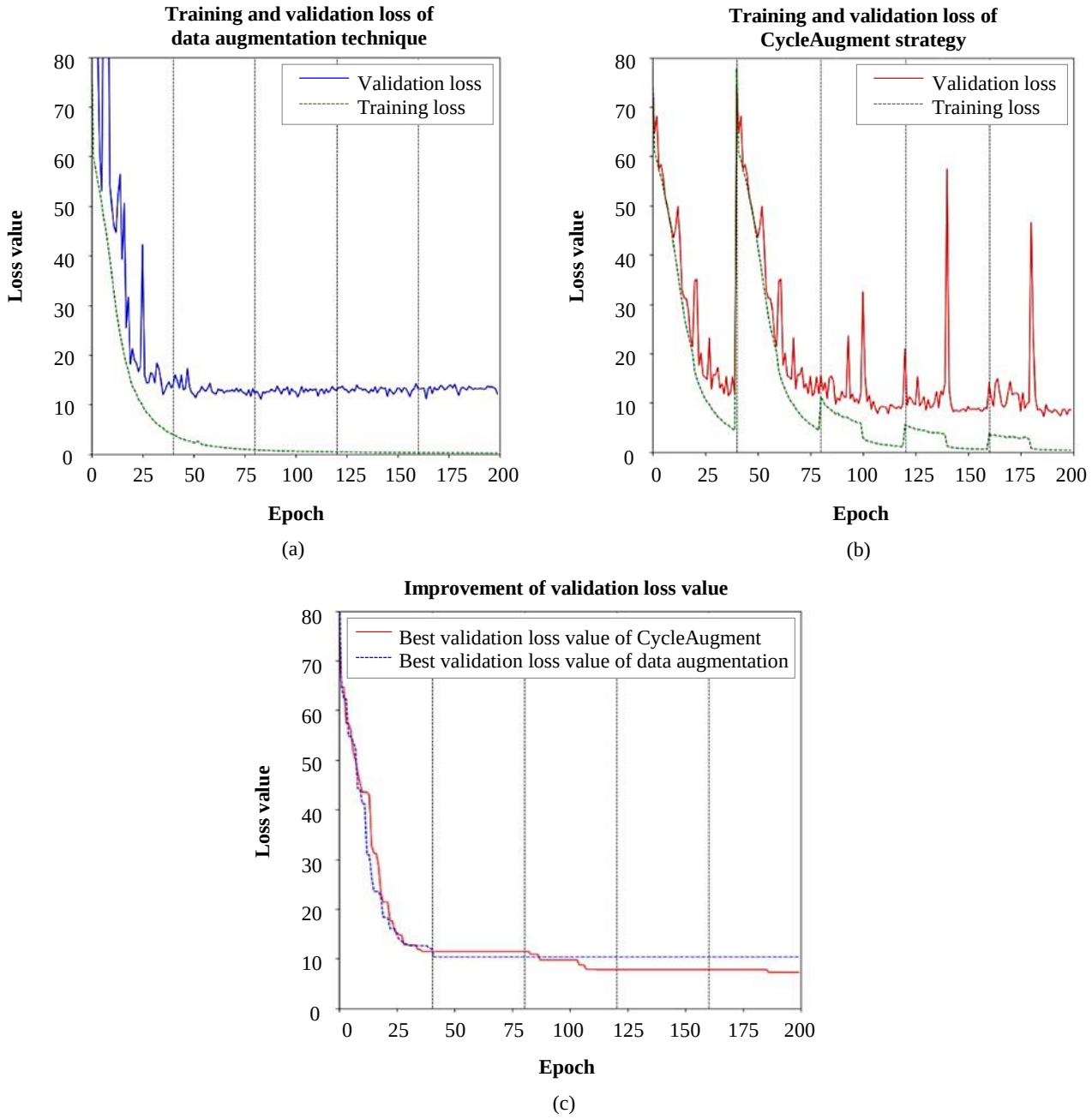
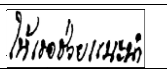
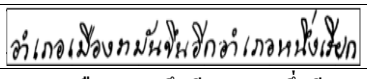


Figure 6 Illustration of the training loss and validation loss values of (a) original data augmentation technique, (b) CycleAugment strategy, and (c) best loss value

In Figure 6, we illustrate the loss values of two data augmentation strategies. The training and validation loss of the original data augmentation strategy is presented in Figure 6(a). The training loss values reduced smoothly and approached zero, but the validation loss was not reduced below 10. If we trained the model for more than 200 epochs, the validation loss may be increased. On the other hand, loss values of the CycleAugment strategy, as shown in Figure 6(b), were rapidly decreased in the first cycle and then grew up at the increased at the beginning of the next cycle. Because, firstly, the CRNN model learns without applying the data augmentation technique, so the CRNN model tries to converge for that particular pattern. Secondly, the CRNN attempts to fit the model with the new input data when applying the data augmentation techniques. Since the model was never trained with the new data, that is why the loss value increased but then quickly decreased again. We then showed the performance of the CycleAugment strategy compared to the original data augmentation strategy, as shown in Figure 6(c). Furthermore, when we trained more epochs, the loss value still could slowly decrease, while the loss value of the original data augmentation strategy stopped decreasing from around epoch 40. This demonstrates that the CycleAugment strategy could benefit from learning with different patterns and avoiding overfitting problems.

Table 7 Results of handwritten text recognition using different CRNN models

	Input image		
	Ground truth	**ให้เธอช่วยแนะนำ**	**อำเภอเมืองรามันขึนอีกอำเภอหนึ่งเรียก**
No data augmentation	mVGG16-BiLSTM (128)	ให้เธอช่วยแนะนำ	อำเภอเมืองรามันจินอีก**อำเภอ**หน่งเรียก
	mVGG19-BiGRU (128)	ให้เธอช่วยแนะ**ห**นำ**ก**	อำ**เก**ภอเมืองรามัน**จิ**นอีกอำเภ**า**อห**นี่ย**งเรียก
	mResNet50-BiLSTM (256)	ให้เธอช่วยแนะนำ	อำเภอเมืองรามัน**จ้**นอีกอำเภอหนึ่งเรียก
	mDenseNet121-BiLSTM (256)	ให้เธอช่วยแนะนำ	อำเภอเมืองรามัน**จิ**่นอีกอำเภอหนึ่งเรียก
Data augmentation	mVGG16-BiLSTM (128)	ให้เธอช่วยแนะนำ	อำ**เถ**อเมืองรามัน**ขิ**นอีกอำเภอหนึ่งเรียก
	mVGG19-BiGRU (128)	ให้เธอ**ช้**วยแนนนำ	อำเภอเมืองราม**นิ**น**จีดชำเราจ**หนึ่ง**มี**ยก**ด**
	mResNet50-BiLSTM (256)	ให้เธอช่วยแนะนำ	อำเภอเมือง**ถ**ามันขึนอีกอำ**เ**ภอหนึ่งเรียก
	mDenseNet121-BiLSTM (256)	ให้เธอช่วยแนะนำ	อำเถอเมืองราม**จี**นอีกอำเภอหนึ่งเรียก
Cycle Augment	mVGG16-BiLSTM (128)	ให้เธอช่วยแนะนำ	อำเภอเมืองรามันขึนอีกอำเภอหนึ่งเรียก
	mVGG19-BiGRU (128)	โ้เอช่**อ**ยแนะนำ	อำเภ**ว**เมือง**ท**มัน**จ**นอีก**ตำ**เภอหนึ่งเ**ือ**ยก
	mResNet50-BiLSTM (256)	ให้เธอช่วยแนะนำ	อำเภอเมืองรามันขึนอีกอำเภอหนึ่งเรียก
	mDenseNet121-BiLSTM (256)	ให้เธอช่วยแนะนำ	อำเภอเมือง**ท**ามัน**ขี่**นอีกอำเภอหนึ่งเรียก

*Note that **bold characters with an underline** represent error characters.

In Table 7, the green text means the CRNN model recognizes and obtains correct output with the whole words. The blue characters with underlining are misclassified characters.

4.6 Performance on short word recognition

In previous experiments, we focused on the performance of the handwritten text dataset consisting of various distributions of word length. Figure 7 presents the histogram of image width resolution in pixels on (a) the whole dataset and (b) only the test set. The image width is distributed in the range of 36 to 1,075 pixels. Due to various ranges of the image width, we are curious whether the short word images (image width between 36 to 186 pixels) affect the recognition performance.

We evaluated the performance of the short word by selecting the short word images from the test set. First, we resized the short word into 64x496 pixels and then recognized the short word images. Second, we resized the short word images into 64x346 pixels and then added white space on the left (75 pixels) and right (75 pixels) sides to prevent the distortion of the texts in the image. Hence, the input image was equal to 64x496 pixels.

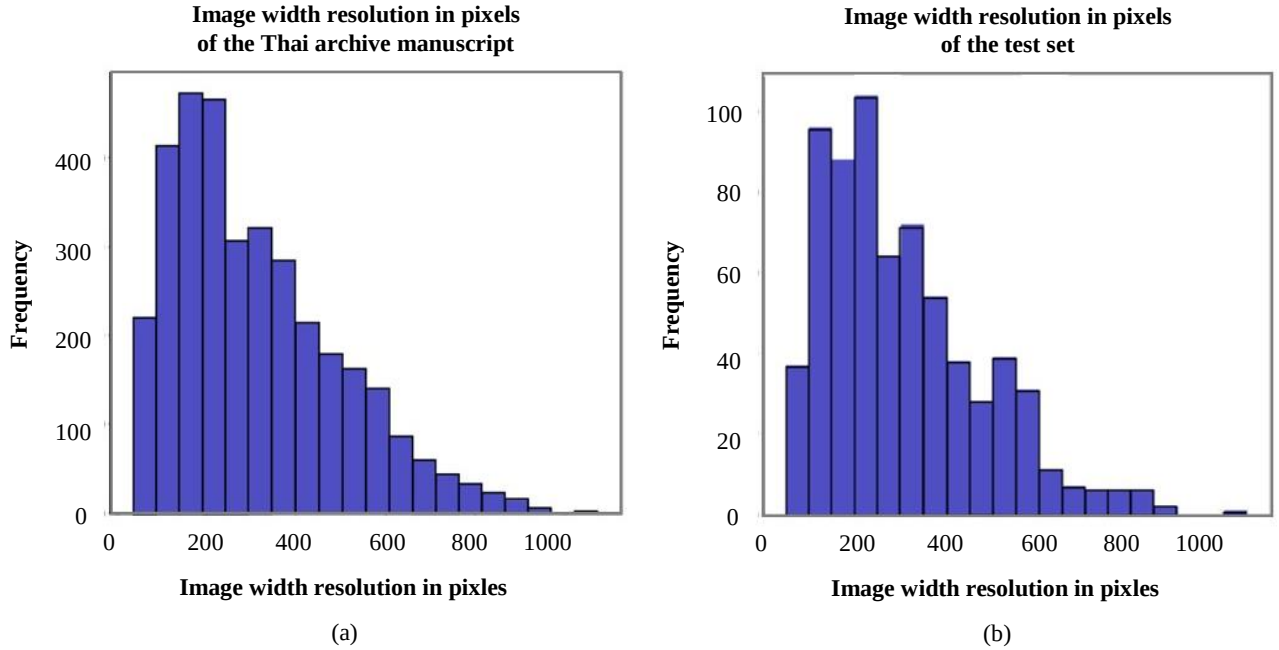
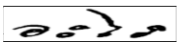
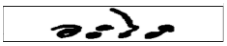


Figure 7 Illustrated histograms of the image width resolution in pixels. (a) The Thai archive manuscript and (b) test set of the Thai archive manuscript

We examined the short word performance using mResNet50-BiLSTM (256) model. Firstly, we evaluated the performance of the short word images (see Table 8 in the second column) and achieved the CER value of 8.07%. Secondly, the short word images were adjusted by adding the white space (see Table 8 in the third column). The experimental results showed that adding the white space before sending to predict by the CRNN model resulted in better performance than did resizing the image. It achieved a CER value of 7.04%. Consequently, we found that image distortion could harm the handwritten text recognition system. It was necessary to rescale the short word images and combine a space into the images before recognizing them.

Table 8 Examples of short word recognition when resizing images into 64x496 pixels (second column) and adding white space to prevent image distortion (third column)

Original image resolution	**Testing image resolution**	**Adjusting image resolution**
		
Ground truth = ดู	Prediction = ค1	Prediction = ค
		
Ground truth = ๓	Prediction = ก1	Prediction = ๓
		
Ground truth = อะไร	Prediction = อฐโร	Prediction = อ๐ไร
		
Ground truth = เมื่อ	Prediction = เมือ	Prediction = เมื่อ
All short word test images	CER = 8.07%	CER = 7.04%

*Note that characters with an underline represent error characters.

5. Discussion

5.1 CycleAugment strategy

It is known that deep learning requires data augmentation techniques to improve performance and avoid overfitting problems. To create the robust CRNN model, we then applied the data augmentation technique. The experimental results showed that the data augmentation techniques did not always confirm the best performance. Consequently, the CRNN model will find only the global minima value when training the CRNN with the original data augmentation strategy. The training loss never again increases, as shown in Figure 6(a). Indeed, it increases the chance of encountering overfitting problems.

We then proposed the new cyclical learning method, namely the CycleAugment strategy. The proposed strategy can effectively improve the performance of the handwritten text recognition by escaping the trapping in global minima and overfitting problems. The CycleAugment strategy increases the chances of discovering local minima in each cycle by switching between two training states with and without applying data augmentation while training the CRNN model, as shown in Figure 6(b). The CRNN model adapted to the local minima because the weight of the CRNN architecture is adjusted using a high error gradient value obtained from variation of the input images.

5.2 Effectiveness of transfer learning technique

We have learned from many studies that the transfer learning technique consistently performed better than scratch learning [31, 32, 34]. Therefore, we evaluated the performance of the scratch and transfer learning, as shown in Table 5 and 6. The experimental results were quite surprising in that the transfer learning performance did not significantly outperform the scratch learning. However, in the CRNN architecture, we discovered that the transfer learning did not show outstanding results because the number of transfer parameters from the pre-trained CNN model was more limited than the parameters in the RNN architecture. We have to train the RNN model with a huge number of parameters that did not transfer from the pre-trained model. The parameters of the RNN architecture are larger, approximately four times more than the CNN architecture.

5.3 Improvement of short word recognition

We also observed that short word images directly decrease the performance of the handwritten text recognition system, as shown in Table 8. We found that the short word images were always distorted when resizing to the fixed input of the CRNN architecture. Hence, we employed the most straightforward technique that avoids distortion of text information in short word images. The simple technique is to adjust the short word images by adding white space on both sides of the image. The performance was presented when applying our proposed method.

6. Conclusion

In recent years, some research has attempted to address the challenge of the Thai handwritten text recognition system. This study discovered a robust CRNN architecture, a sequence learning approach that achieves high accuracy on the Thai handwritten text recognition system. For training the CRNN model, the original data augmentation strategy is proposed. The CRNN model was trained by applying the transformation data augmentation techniques from the first training epoch until the last epoch. With this training strategy, the CRNN model slowly obtained the global minima value. The model can face overfitting problems because the training loss decreases to the lowest value. However, the validation loss sometimes does not converge to the lowest value. Nonetheless, we invented a cyclical data augmentation strategy called CycleAugment, to avoid finding the global minima and control overfitting problems. In our strategy, all training epochs are divided into cycles. In each cycle, we assign the CRNN model to discover the local minimal value. Hence, it repeatedly starts at high loss value by learning new patterns from the training images when beginning a new cycle. As a result, the weight model is adapted by a high gradient value. The benefit of our proposed CycleAugment strategy is that the CRNN model can learn from both with and without applying data augmentation techniques.

In the experiments, we evaluated nine CRNN architectures to recognize handwritten text on the Thai archive manuscript dataset. The result showed that the mDenseNet121-BiLSTM (256) outperformed all the CRNN architectures. First, we performed the CRNN architectures using scratch and transfer learning. It is quite surprising that transfer learning did not show a significant performance when compared with scratch learning. Second, we trained the CRNN models with three different data augmentation strategies: without data augmentation, with data augmentation, and CycleAugment. The proposed CycleAugment strategy achieved the best performance when combined with all CRNN models. Finally, we are concerned about the performance of the CRNN model when predicting the short word images. The text information inside the short word images is regularly distorted when transformed into the input of the CRNN model with the same size as the long word images. We proposed the simple technique is of adding white space on both sides of the short word images. We achieved a better result with the simple technique.

In future work, due to insufficient handwritten text images for training the CRNN model, the model might not give generalizations. We might need to synthesize the handwritten text images and use them as the training set. The generative adversarial network (GAN) [47] is the best choice to study and synthesize the training set. In sequential learning, we need to investigate an attention-based model [22, 23, 48] and word beam search [14] to better predict the handwritten text images.

7. Acknowledgments

This research was funded under the Royal Golden Jubilee Ph.D. Program by the Thailand Research Fund (Grant No. PHD/0210/2561).

8. References

[1] Surinta O, Karaaba MF, Schomaker LRB, Wiering MA. Recognition of handwritten characters using local gradient feature descriptors. Eng Appl Artif Intell. 2015;45:405-14.
[2] Inkeaw P, Bootkrajang J, Marukatat S, Gonçalves T, Chaijaruwanich J. Recognition of similar characters using gradient features of discriminative regions. Expert Syst Appl. 2019;134:120-37.
[3] Wang T, Xie Z, Li Z, Jin L, Chen X. Radical aggregation network for few-shot offline handwritten Chinese character recognition. Pattern Recognit Lett. 2019;125:821-7.
[4] Kavitha BR, Srimathi C. Benchmarking on offline handwritten Tamil character recognition using convolutional neural networks. J King Saud Univ Comp Info Sci. In press 2019.
[5] Choudhary A, Rishi R, Ahlawat S. A new character segmentation approach for off-line cursive handwritten words. Procedia Comput Sci. 2013;17:88-95.
[6] Lue HT, Wen MG, Cheng HY, Fan KC, Lin CW, Yu CC. A novel character segmentation method for text images captured by cameras. ETRI J. 2010;32(5):729-39.
[7] Inkeaw P, Bootkrajang J, Charoenkwan P, Marukatat S, Ho SY, Chaijaruwanich J. Recognition-based character segmentation for multi-level writing style. Int J Doc Anal Recognit. 2018;21(1-2):21-39.
[8] Giménez A, Juan A. Embedded Bernoulli mixture HMMs for handwritten word recognition. 10th International Conference on Document Analysis and Recognition; 2009 Jul 26-29; Barcelona, Spain. New York: IEEE; 2009. p. 896-900.
[9] Bluche T, Ney H, Kermorvant C. Feature extraction with convolutional neural networks for handwritten word recognition. 12th International Conference on Document Analysis and Recognition; 2013 Aug 25-28; Washington, USA. New York: IEEE; 2013. p. 285-9.
[10] Wang K, Babenko B, Belongie S. End-to-end scene text recognition. International Conference on Computer Vision; 2011 Nov 6-13; Barcelona, Spain. New York: IEEE; 2011. p. 1457-64.
[11] Lee C, Bhardwaj A, Di W, Jagadeesh V, Piramuthu R. Region-based discriminative feature pooling for scene text recognition. IEEE Conference on Computer Vision and Pattern Recognition; 2014 Jun 23-28; Columbus, USA. New York: IEEE; 2014. p. 4050-7.
[12] Mishra A, Alahari K, Jawahar CV. Enhancing energy minimization framework for scene text recognition with top-down cues. Comput Vis Image Underst. 2016;145:30-42.
[13] Singh S, Sharma A, Chauhan VK. Online handwritten Gurmukhi word recognition using fine-tuned deep convolutional neural network on offline features. Mach Learn with Appl. 2021;5(article 100037):1-15.
[14] Ameryan M, Schomaker L. A limited-size ensemble of homogeneous CNN/LSTMs for high-performance word classification. Neural Comput Appl. 2021;33:8615-34.
[15] Chen Y, Shu H, Xu W, Yang Z, Hong Z, Dong M. Transformer text recognition with deep learning algorithm. Comput Commun. 2021;178:153-60.
[16] Abdurahman F, Sisay E, Fante KA. AHWR-Net: offline handwritten Amharic word recognition using convolutional recurrent neural network. SN Appl Sci. 2021;3(760):1-11.
[17] Yan H, Xu X. End-to-end video subtitle recognition via a deep residual neural network. Pattern Recognit Lett. 2020;131:368-75.

[18] Sujatha P, Bhaskari DL. A survey on offline handwritten text recognition of popular Indian scripts. Int J Comput Sci Eng. 2019;7(7):138-49.
[19] Butt H, Raza MR, Ramzan MJ, Ali MJ, Haris M. Attention-based CNN-RNN Arabic text recognition from natural scene images. Forecasting. 2021;3(3):520-40.
[20] Huang G, Li Y, Pleiss G, Liu Z, Hopcroft JE, Weinberger KQ. Snapshot ensembles: train 1, get M for free. International Conference on Learning Representations (ICLR); 2017 Apr 24-26; Toulon, France. New York: DBLP; 2017. p. 1-14.
[21] 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. 2017;39(11):2298-304.
[22] Shi B, Yang M, Wang X, Lyu P, Yao C, Bai X. ASTER: an attentional scene text recognizer with flexible rectification. IEEE Trans Pattern Anal Mach Intell. 2018;41(9):2035-48.
[23] Luo C, Jin L, Sun Z. A multi-object rectified attention network for scene text recognition. Pattern Recognit. 2019;90:109-18.
[24] Chen X, Jin L, Zhu Y, Luo C, Wang T. Text recognition in the wild: a survey. ACM Comput Surv. 2021;54(2):1-35.
[25] Xu Y, Shan S, Qiu Z, Jia Z, Shen Z, Wang Y, et al. End-to-end subtitle detection and recognition for videos in East Asian languages via CNN ensemble. Signal Process Image Commun. 2018;60:131-43.
[26] Chamchong R, Gao W, McDonnell MD. Thai handwritten recognition on text block-based from Thai archive manuscripts. International Conference on Document Analysis and Recognition (ICDAR); 2019 Sep 20-25; Sydney, Australia. New York: IEEE; 2019. p. 1346-51.
[27] Srinilta C, Chatpoch S. Multi-task learning and Thai handwritten text recognition. 6th International Conference on Engineering, Applied Sciences and Technology (ICEAST); 2020 Jul 1-4; Chiang Mai, Thailand. New York: IEEE; 2020. p. 1-4.
[28] Chamchong R, Saisangchan U, Pawara P. Thai handwritten recognition on BEST2019 datasets using deep Learning. International Conference on Multi-disciplinary Trends in Artificial Intelligence (MIWAI); 2021 Jul 2-3. Cham: Springer; 2021. p. 152-63.
[29] Wu B, Liu Z, Yuan Z, Sun G, Wu C. Reducing overfitting in deep convolutional neural networks using redundancy regularizer. 26th International Conference on Artificial Neural Networks (ICANN); 2017 Sep 11-14; Alghero, Italy. Cham: Springer; 2017. p. 49-55.
[30] Thanapol P, Lavangnananda K, Bouvry P, Pinel F, Leprévost F. Reducing overfitting and improving generalization in training convolutional neural network (CNN) under limited sample sizes in image recognition. 5th International Conference on Information Technology (InCIT); 2020 Oct 21-22; Chonburi, Thailand. New York: IEEE; 2020. p. 300-5.
[31] Gonwirat S, Surinta O. Improving recognition of Thai handwritten character with deep convolutional neural networks. International Conference on Information Science and Systems (ICISS); 2020 Mar 19-22; Cambridge, UK. New York: Association for Computing Machinery; 2020. p. 87-7.
[32] Pawara P, Okafor E, Surinta O, Schomaker L, Wiering M. Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition. International Conference on Pattern Recognition Applications and Methods (ICPRAM); 2017 Feb 24-26; Porto, Portugal. Setúbal: SciTePress; 2017. p. 479-86.
[33] Pawara P, Okafor E, Schomaker L, Wiering M. Data augmentation for plant classification. Advanced Concepts for Intelligent Vision Systems (ACIVS); 2017 Sep 18-21; Antwerp, Belgium. Cham: Springer; 2017. p. 615-26.
[34] Enkvetchakul P, Surinta O. Effective data augmentation and training techniques for improving deep learning in plant leaf disease recognition. Appl Sci Eng Prog. 2021;15(3):1-12.
[35] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations (ICLR); 2015 May 7-9; San Diego, USA. New York: DBLP; 2015. p. 1-14.
[36] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jul 27-30; Las Vegas, USA. New York: IEEE; 2016. p. 770-8.
[37] Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely connected convolutional networks. Conference on Computer Vision and Pattern Recognition (CVPR); 2017 Jul 21-26; Honolulu, USA. New York: IEEE; 2017. p. 2261-9.
[38] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. MobileNetV2: inverted residuals and linear bottlenecks. Conference on Computer Vision and Pattern Recognition (CVPR); 2018 Jun 18-23; Salt Lake City, USA. New York: IEEE; 2018. p. 4510-20.
[39] Tan M, Le QV. EfficientNet: rethinking model scaling for convolutional neural networks. International Conference on Machine Learning (ICML); 2019 Jun 9-15; Long Beach, California. New York: PMLR; 2019. p. 6105-14.
[40] Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks. International Conference on Machine Learning (ICML); 2014 Jun 21-26; Beijing, China. New York: PMLR; 2014. p. 1764-72.
[41] Donahue J, Anne Hendricks L, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K, et al. Long-term recurrent convolutional networks for visual recognition and description. IEEE Trans Pattern Anal Mach Intell. 2017;39(4):677-91.
[42] Alhagry S, Fahmy AA, El-Khoribi RA. Emotion recognition based on EEG using LSTM recurrent neural network. Int J Adv Comput Sci Appl. 2017;8(10):355-8.
[43] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735-80.
[44] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. Conference on Empirical Methods in Natural Language Processing; 2014 Oct 25-29; Doha, Qatar. USA: Association for Computational Linguistics; 2014. p. 1724-34.
[45] Graves A, Fernández S, Gomez F, Schmidhuber J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. International Conference on Machine Learning (ICML); 2006 Jun 25-29; Pittsburgh, USA. New York: Association for Computing Machinery; 2006. p. 369-76.
[46] Bluche T. Deep neural networks for large vocabulary handwritten text recognition networks for large vocabulary handwritten text recognition [thesis]. Orsay: Universite´ Paris Sud-Paris XI; 2015.
[47] Fogel S, Averbuch-Elor H, Cohen S, Mazor S, Litman R. ScrabbleGAN: semi-supervised varying length handwritten text generation. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020 Jun 13-19; Seattle, USA. New York: IEEE; 2020. p. 4323-32.
[48] Atienza R. Vision transformer for fast and efficient scene text recognition. In: Lladós J, Lopresti D, Uchida S, editors. Document analysis and recognition-ICDAR 2021. Cham: Springer; 2021. p. 319-34.