Khmer printed character recognition using attention-based Seq2Seq network

Rina Buoy[1*], Nguonly Taing[1], Sovisal Chenda[1], Sokchea Kor[2]

[1]Techo Startup Center, Phnom Penh, Cambodia
[2]Royal University of Phnom Penh, Phnom Penh, Cambodia
[*]Corresponding author: rinabuoy13@gmail.com

ARTICLE INFO

DOI:10.46223/HCMCOUJS.tech.en.12.1.2217.2022

Received: March 28th, 2022

Revised: April 09th, 2022

Accepted: April 15th, 2022

Keywords:

character recognition; deep learning; Khmer; optical neural network

ABSTRACT

This paper presents an end-to-end deep convolutional recurrent neural network solution for Khmer Optical Character Recognition (OCR) task. The proposed solution uses a sequence-to-sequence (Seq2Seq) architecture with an attention mechanism. The encoder extracts visual features from an input text-line image via layers of convolutional blocks and a layer of Gated Recurrent Units (GRU). The features are encoded in a single context vector and a sequence of hidden states which are fed to the decoder for decoding one character at a time until a special End-Of-Sentence (EOS) token is reached. The attention mechanism allows the decoder network to adaptively select relevant parts of the input image while predicting a target character. The Seq2Seq Khmer OCR network is trained on a large collection of computer-generated text-line images for multiple common Khmer fonts. Complex data augmentation is applied to both the train and validation datasets. The proposed model's performance outperforms the state-of-art Tesseract OCR engine for the Khmer language on the validation set of 6,400 augmented images by achieving a Character Error Rate (CER) of 0.7% vs. 35.9%.

1. Introduction

One of the Artificial Intelligence (AI) paradigms is to develop a machine that can mimic the ability of human recognition. In terms of visual perception and understanding text, the computer is still at the infancy level compared with humans (Annanurov & Noor, 2018). To make a text machine-readable, it can be either converted manually or digitally extracted by means of Optical Character Recognition (OCR) from the digital image of the document (Memon, Sami, Khan, & Uddin, 2020). OCR is the science of extracting analyzable and editable data from scanned documents or images. The OCR technology has been evolving over the last 08 decades. Large tech players mainly contributed to the early phase of OCR development. The recent advancement of artificial intelligence, particularly deep learning, has allowed researchers from various spectrums to devise OCR algorithms that can achieve higher accuracy levels (Memon et al., 2020). Although OCR technology for English and other high-resource languages has been developed over the last 08 decades (Memon et al., 2020), the early OCR work on the Khmer language was around the year 2005.

Khmer (KHM) is the official language of the kingdom of Cambodia. The Khmer script is used in the writing system of Khmer and other minority languages such as Kuay, Tampuan, Jarai Krung, Brao, and Kravet. Khmer language and writing system were hugely influenced by Pali and

Sanskrit in early history (Bahdanau, Cho, & Bengio, 2014; Buoy, Taing, & Kor, 2021; Sok, 2016). Unlike Latin-based languages, the Khmer language has a complex writing system. One or two consonants can be stacked below an initial consonant using the alternate form (aka Coeng - foot in English) to form a consonant cluster (Bahdanau et al., 2014; Sok, 2016). Khmer writing also uses diacritical signs which are placed above a consonant. Dependent vowels cannot stay alone by themselves and must be attached to an initial consonant. Orthographically, a dependent vowel can be placed to the left, right, above, below, or around a base consonant (Ding, Utiyama, & Sumita, 2018). Therefore, Khmer scripts require a complex rendering layout, which is not the case with Latin-based writing systems. A complete Khmer OCR system needs to recognize all characters, given the complexity of Khmer writing.

Khmer is a low-resource language in Natural Language Processing (NLP) context, and research on Khmer OCR tasks is still limited, although OCR is one of the fundamental NLP tasks with many practical applications. Therefore, a robust Khmer OCR is required. Recent advances in artificial intelligence and specifically deep learning have made it possible to train OCR models in an end-to-end fashion without complex pre-processing or post-processing.

Research works on Khmer OCR have primarily focused on using complex feature extraction steps, traditional machine learning classifiers, and post-processing steps. Such approaches are difficult to optimize simultaneously and do not yield an acceptable accuracy level. Other solutions are not complete as they can predict only standalone characters instead of full words, phrases, or sentences. Therefore, there requires an end-to-end solution to the Khmer OCR task which can read a raw text-line image of any length and outputs editable text in a single forward run. The solution to the Khmer OCR task should recognize texts with different fonts and in different environments.

The primary objectives of this work are:

• To develop an end-to-end OCR pipeline for multi-font Khmer text recognition utilizing a deep learning-based sequence-to-sequence model with an attention mechanism. An end-to-end OCR integrates feature extraction, classification, and post-processing in a single network, which can be optimized simultaneously.

• To achieve the state-of-art performance (SOTA) in Khmer text recognition.

2. Literature review

2.1. Khmer Optical Character Recognition (OCR)

Chey, Kumhom, and Chamnongthai (2005) did one of the early works in Khmer OCR. The proposed method was a variant of instance-based classifiers. The authors used wavelet descriptors to extract features (coefficients) from pre-processed images in the training set and built a template for each character. For a given new input image, a set of wavelet coefficients was extracted and matched against all the training templates. The input image was then assigned to the class with the smallest Euclidean distance. The classifier could classify only images with a standalone character and was not scale-invariant. The highest accuracy was 92.99% at 300 dpi resolution.

Sok and Taing (2014), Chey et al. (2005) applied the Support Vector Machine (SVM) algorithm to recognize Khmer characters. The complete pipeline was composed of four steps - character segmentation, feature extraction, classification, and character reassembling. The reported character classification accuracy was about 98% for various font sizes. The system performance was dependent on the character segmentation step, which relied on edge detection. The proposed OCR system was, therefore, not applicable to noisy text-line images.

Meng and Morariu (2014) applied an artificial neural network in recognizing Khmer characters. The approach is applied to standalone character recognition. The authors proposed two steps recognition pipeline. An input image (20 by 20 pixels) was first passed into a self-organizing network, which grouped the input image into one of the nine classes. Each class had one multi-layer neural network, which classified the input image to one of 82 Khmer characters, including consonants, vowels, and numbers. The average recognition rate on the train and test datasets was 65% and 30%, respectively.

Lenleng and Muaz (2015) from the PAN Localization Cambodia team proposed a complete workflow for recognizing Khmer text. Similar to Sok and Taing (2014), Chey et al. (2005), the four-step workflow included pre-processing, segmentation, recognition, and mapping. The pre-processing step included line separation and character block segmentation. Blocks of characters were then segmented into atomic shapes, namely: Main Body, Super-Script, SubScript, CCDown, and CC (Complex Character). Discrete cosine transform was used to extract features from the atomic shapes for the classification task. The recognized shapes were finally mapped to produce valid Khmer text. The average recognition rate for all shapes was reported to be 96.34%. The system performance was dependent on character separation, which used a vertical white space as a delimiter. The proposed OCR system was, therefore, not applicable to noisy text-line images.

Valy, Verleysen, Chhun, and Burie (2017) proposed a character-level Convolution Neural Network (CNN) classifier in recognizing ancient Khmer characters on palm-leaf manuscripts. The proposed CNN architecture was applied to standalone character recognition and is composed of 03 convolutional blocks and a linear classifier. The classifier output a vector of 106 elements representing character classes. The accuracy of the test set was reported at around 95.96.

Annanurov and Noor (2018) experimented with both multi-layer and convolutional neural networks to recognize standalone Khmer consonants. A CNN-based model was compared against Artificial Neural Network (ANN)-based classifier with a full feature set and an ANN-based classifier with a reduced feature set. The CNN model achieved up to 94.85% average accuracy. However, the model could recognize standalone Khmer consonants only.

Sokphyrum, Samak, and Sola (2019) fined tune a pre-trained Tesseract OCR engine for Khmer Unicode and legacy Lemon fonts. Tesseract is an end-to-end multilingual OCR engine. Tesseract uses deep convolutional recurrent neural network architecture with Connectionist Temporal Classification (CTC) loss. Tesseract learns feature representation automatically via one convolutional layer followed by multiple stacked recurrent neural networks (Liebl & Burghardt, 2020). Tesseract can recognize a text-line image. Sokphyrum et al. (2019) reported an accuracy of 90% on the fine-tuned fonts.

2.2. Seq2Seq Network and Attention

Although Deep Neural Networks (DNNs) are very good models in computer vision or natural language processing, DNNs are not able to handle inputs and targets of variable lengths. This limitation prevents DNNs from being applied to certain tasks, such as speech recognition and machine translation, in which sequence lengths are not fixed (Sutskever, Vinyals, & Le, 2014).

Recurrent Neural Networks (RNNs), on the other hand, can encode an input sequence of unknown length to produce a fixed-dimension representation of the input sequence. Thus, Sutskever et al. (2014) proposed an RNN known as an encoder to encode the input sequence of variable length to a fixed-dimension vector representation, commonly known as a context vector. The context vector was passed to another RNN known as a decoder to extract the information and generate the output sequence at any length. An encoder-decoder network, also known as Seq2Seq, was applied to the English-French machine translation task. A BLEU score of 34.81 was achieved,

which was, at that time, the best result by neural machine translation. The authors made the observation that the Long Short-Term Memory (LSTM) architecture performed robustly even on long sequences, although the previous study suggested otherwise.

Bahdanau et al. (2014) suggested that using a single fixed-dimension context vector is a bottleneck for the encoder-decoder network. To remove this bottleneck, the attention mechanism was introduced to allow the decoder to selectively use only parts of the input sequence that are useful for predicting a target word.

The key distinction between Sutskever et al. (2014) and Bahdanau et al. (2014) is that the former proposed a single fixed-length context vector for the whole input sentence, while the latter proposed to encode the input sequence as a sequence of hidden vectors on top of the context vector and the decoder chooses a subset of these vectors adaptively via attention mechanism while decoding the translation. The experimental results from Bahdanau et al. (2014) suggested that the proposed approach outperformed the conventional encoder-decoder significantly on the English-to-French translation task.

OCR task can be viewed as another machine translation task. The input to the OCR systems is an input image of a text line with variable length. The output of the OCR systems is a character sequence of variable length. Therefore, the solutions to machine translation tasks can also be extended to OCR tasks.

Sahu and Sukhwani (2015) proposed an end-to-end encoder-decoder network for recognizing printed text. Long Short-Term Memory (LSTM) networks were used in both encoder and decoder. The label error rate of 0.84% was reported on the annotated English word images.

Safir, Ohi, Mridha, Monowar, and Hamid (2021) proposed an end-to-end convolutional recurrent neural network with Connectionist Temporal Classification (CTC) loss for Bengali handwritten words. Safir et al. (2021) experimented with various CNN feature extractors such as DenseNet, Xception, NASNet, and MobileNet and different recurrent neural networks such as LSTM and GRU. Safir et al. (2021) reported 0.091 Character Error Rate (CER) and 0.273 Word Error Rate (WER) by using the DenseNet121 model with GRU recurrent layer.

In this work, we attempt to approach the Khmer OCR task by using an encoder-decoder network. We propose an end-to-end, attention-based encoder-decoder network (Seq2Seq). The encoder is a convolutional recurrent neural network, while the decoder is a recurrent neural network followed by a linear classifier. During decoding, the attentive decoder uses an attention mechanism to search for the relevant encoder outputs. Gated Recurrent Unit (GRU) is used in both the decoder and encoder. The entire OCR network can read a text-line image of arbitrary length and produces a character sequence of unknown length. Being an end-to-end solution, the proposed solution does not need any pre-processing, feature extraction, or post-processing steps.

3. An end-to-end Khmer OCR network

We propose the end-to-end Khmer OCR model using a Seq2Seq (aka encoder-decoder) network with an attention mechanism. The end-to-end OCR model does not require pre-processing steps such as character separation and post-processing steps such as character mapping or reassembling.

The proposed Seq2Seq network takes a text-line input image via the encoder network. The encoder network passes the encoded information via a context vector and a sequence of hidden vectors to the decoder network to decode one character at a time until the End-Of-Sentence (EOS) token is reached. There is no limit on the number of characters in the input text-line image that the encoder can read and the decoder can produce. The attention mechanism allows the network performs robustly even with an input image with a very long text line. The high-level architecture

of the end-to-end, attention-based Seq2Seq network for Khmer OCR is given in Figure 1. Figure 1 shows the decoding process of the second time step while the decoding process of the first time step is being grayed out. Detailed description of the encoder, decoder and attention is provided in the subsequent sections.

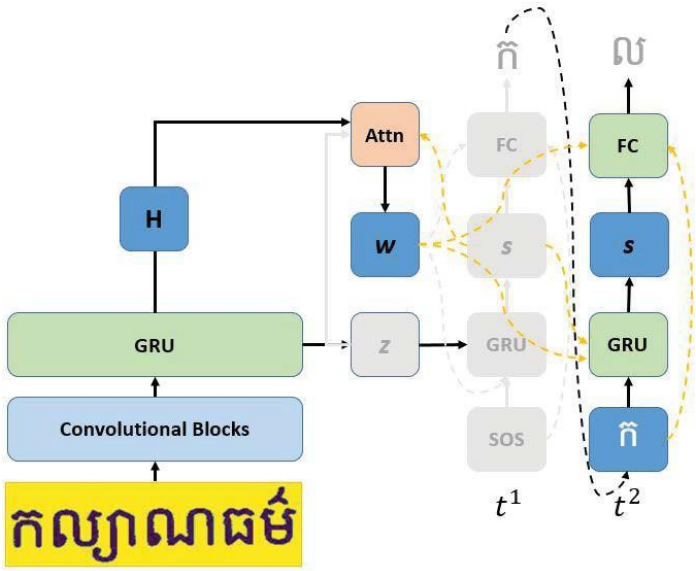

Figure 1. The proposed end-to-end, attention-based Seq2Seq network for Khmer OCR

3.1. Encoder

The encoder is composed of two parts: convolutional blocks and recurrent layers. As illustrated in Figure 2, the role of the convolution blocks is to produce a sequence of feature maps from a given input text-line image. A sequence of feature maps is proportional to the input image's width. The sequence of feature maps of *T* length is further fed to the bidirectional recurrent layers, which are Gated Recurrent Units (GRU) in this case. The recurrent layers produce two context vectors (i.e., last hidden states) which are backward, $h_{\overleftarrow{T}}$ and forward, $h_{\overrightarrow{T}}$. Both context vectors are concatenated to produce a single context vector, h_T which is fed to a fully connected layer and followed by an arc-tan activation function. At each step in the sequence, the encoder also outputs a concatenated hidden vector. Therefore, the encoder outputs a transformed, squashed context vector, *z*, as well as a sequence of concatenated hidden vectors, $H = \{h_1, h_2, \ldots, h_T\}$. The decoder takes *z* as its initial state, so and the hidden states, *H*.

The detailed specification of the encoder is given in Figure 3.

3.2. Attention

At each decoding step *i*, the decoder needs to search the relevant information from the encoder's hidden states (i.e., vectors) adaptively. This can be done by calculating a weighted average context vector, ci, over the encoder's hidden states (Bahdanau et al., 2014).

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} * h_j \tag{1}$$

where:

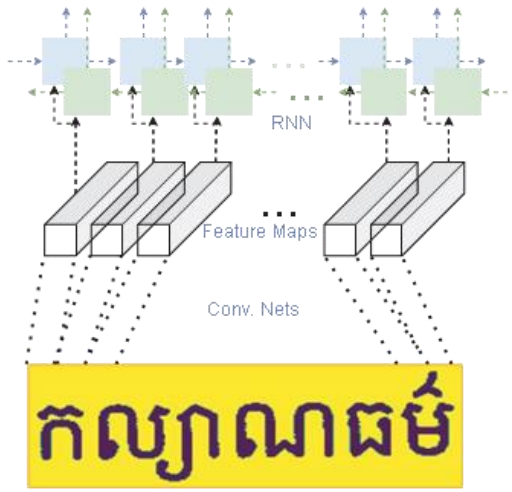

Figure 2. The encoder

• T_x is the length of the input sequence;

• c_i is the weighted average context vector at decoding step I;

• α_{ij} is the attention weight for the encoder's hidden state at time j and the decoder's previous hidden state at the time I;

• h_j is the encoder's hidden state at time j.

α_{ij} can be computed using the below formula (Bahdanau, Cho, & Bengio, 2014).

$$\alpha_{ij} = \frac{e^{e_{ij}}}{\sum_{k=1}^{T_x} e^{e_{ik}}} \tag{2}$$

$$e_{ij} = v_a^T \tanh(W_a s_{i-1} + U_a h_j) \tag{3}$$

where:

• v_a, W_a, and U_a are weight matrices;

• s_{i-1} is the decoder's previous hidden state;

The graphical illustration of attention during decoding at a time step, *t* is given in Figure 4.

3.3. Decoder network

The decoder network is a single-layer GRU network followed by a linear classifier. The inputs to the GRU network are:

• $d(y_t)$: the previous decoded target that is one-hot encoded. For $t = 1$, y_0 is a special start-of-sentence (SOS) token;

• s_{t-1}: the decoder's previous hidden state;

• c_t: the weighted average context vector at decoding step, *t*.

	Input	W x 32, gray scale image
Convolutional Neural Network (CNN)	Conv. 1	#filter=64, filter=(3,3), s=1, p=1
	MaxPooling 1	filter=(2,2), s =2
	Conv. 2	#filter=128, filter=(3,3), s=1, p=1
	MaxPooling 2	filter=(2,2), s =2
	Conv. 3	#filter=256, filter=(3,3), s=1, p=1
	Conv. 4	#filter=256, filter=(3,3), s=1, p=1
	MaxPooling 3	filter=(2,1), s =1
	Conv. 5	#filter=512, filter=(3,3), s=1, p=1
	Batch Normalization	
	Conv. 6	#filter=512, filter=(3,3), s=1, p=1
	Batch Normalization	
	MaxPooling 4	filter=(2,1), s =1
	Conv. 7	#filter=512, filter=(3,3), s=1, p=0
	Map to sequence	Linear(512,64)
RNN	GRU 1	#hidden=256, bi-directional=true
	GRU 2	#hidden=256, bi-directional=true
	Fully-connected Layer	Linear(512,512)
	Tanh	

Figure 3. The encoder architecture

$d(y_t)$ and c_t are concatenated as a single input vector to the GRU layer. Together with the previous hidden state, s_{t-1} the GRU layer produces a new hidden state, s_t.

$$s_t = DecoderGRU([d(y_t), c_t], s_{t-1}) \quad (4)$$

Next, s_t, $d(y_t)$, and c_t are concatenated as a single input vector to the linear classifier to predict the target, y_{t+1}.

$$y_{t+1} = LinearClassifer([st, ct, d(yt)]) \quad (5)$$

The decoding process is repeated until y_{t+1} is a special end-of-sentence (EOS) token.

The decoder's GRU layer has a hidden dimension of 512 while y_{t+1} is a vector of 117 elements, corresponding to the number of unique Khmer characters which is given in Figure 5.

3.4. Synthetic dataset generation

The dataset is generated by using the text2image tool developed by the Tesseract team for multiple common Khmer fonts. The rendered images are augmented with speckle noise, dilation/erosion, and rotation. The dataset consists of 03 million words, phrases, or sentences. The images for different fonts for the same word are given in Figure 6.

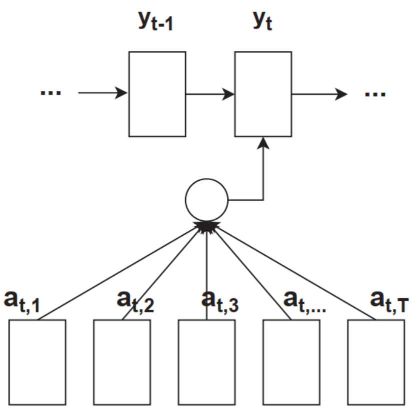

Figure 4. Graphical illustration of attention during decoding (modified from Annanurov and Noor (2018))

Types	Characters
Numbers	០១២៣៤៥៦៧៨៩0123456789
Consonants	កខគឃងចឆជឈញដឋឌឍណតថទធនបផពភមយ រលវឝឞសហឡអឣឤឥឦឧឩឪឫឬឭឮឯឰឱឲឳ
Vowels	◌ា◌ិ◌ី◌ឹ◌ឺ◌ុ◌ូ◌ួ◌ើ◌ឿ◌ៀ◌េ◌ែ◌ៃ◌ោ◌ៅ ◌ំ◌ះ◌ៈ
Subscript	◌្
Diacritics	◌៑ ◌៉◌៊ ◌់ ◌៍ ◌៎ ◌័
Symbols	(),.៖។៕៘ៗ៙៚។ល។?(space)

Figure 5. Khmer characters

Figure 6. Different images of a Khmer word for the different fonts

The text2image generates images with variable width and height, depending on the word or phrase or sentence length and the presence of subscript consonants, vowels, and diacritics. Therefore, the input images are scaled to a common height of 32 pixels, which gives the final height of 01 after the convolutional layers in the encoder network. According to Liebl and Burghardt (2020), Tesseract uses a common height of 48 pixels.

3.5. Data augmentation

The role of data augmentation is to improve model robustness by guiding the model to extract relevant text features and ignore noises (Namysl & Konya, 2019). The generated text-line images are subject to standard text augmentation techniques, which include Gaussian blurring,

dilation, erosion, and blob noise. In addition, fibrous and multi-scale noisy background is also randomly added. Two augmented images can also be randomly concatenated. A probability of 50% is assigned to each augmentation method. Each input image has a 50% chance of being augmented with an augmentation technique. Therefore, more than one augmentation technique can be applied to one input image. Some examples of augmented images are illustrated in Figure 7.

For the training images, data augmentation is applied dynamically on each batch. That means a training image can be augmented more than one time. The validation images, however, are augmented only once. This is to make sure that the same augmented validation images are used for evaluation purposes.

Figure 7. Some examples of augmented images

3.6. Overall workflow

The overall workflow is given in Figure 8. It starts with a text corpus which is a collection of numbers, words, phrases, or sentences. The synthetic images are generated from the text corpus. The generated images are split into two sets – train and validation. One-time data augmentation is applied to the validation set. The train set is used to train the model, and batch-level data augmentation is applied dynamically. The trained model is then evaluated on the validation set and benchmarked against Tesseract-OCR for Khmer on the same validation set.

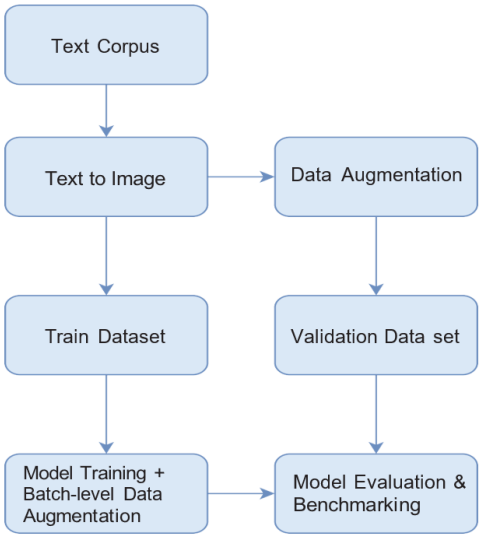

Figure 8. The overall workflow

4. Model training and results

4.1. Training configuration

The model has 9,938,222 trainable parameters. The model's training configurations are given below:

• Epoch: 150;

• Batch size: 64 images;

• Optimizer: Adam;

• Learning rate: 1e-6;

• Loss function: Cross-entropy loss;

• Teacher forcing factor: 1;

• Framework: PyTorch;

• GPU: 1 GPU (Tesla P100-PCIE-16GB on Google Colab Pro).

Due to the restricted run-time usage (~12 hours) by Google Colab, the training had to be restarted from the last checkpoint multiple times.

4.2. Character Rate Error

To measure model performance on the validation set, which consists of 6,400 augmented images, a Character Rate Error (CER) is used. CER is defined as below:

$$CER = \frac{S + D + I}{N} \tag{6}$$

Where:

• S: number of substitutions;

• D: number of deletions;

• I: number of insertions;

• N: number of reference characters The nominator in the CER formula is basically the Levenshtein distance between two strings.

4.3. Model evaluation

The model's CER on the validation set of 6,400 images is 0.7%. Some samples of the augmented validation images are given in Figure 9, which show combined effects of dilation, distortion, noisy background, and noisy blobs.

For evaluation, the pre-trained Tesseract-OCR for Khmer is used to extract texts from the validation images without further fine-tuning. Tesseract 4.00, which uses both convolutional and recurrent neural networks, is used for benchmarking.

The calculated CER of Tesseract-OCR is 35.9%. Namysl and Konya (2019) reported a similar CER by Tesseract-OCR on the highly augmented synthetic images. A more detailed error analysis will be given in the next section.

To assess the effect of data augmentation on the model robustness, the validation set of augmented images is de-augmented. This is illustrated in Figure 10. Then, the CERs are computed for both the proposed solution and Tesseract-OCR. The calculated CERs for the proposed solution and Tesseract-OCR are 0.24% and 1.6%. This suggests perhaps, not enough augmentation was applied for training Tesseract-OCR, although it was otherwise claimed by the Tesseract-OCR team.

4.4. Visualizing attention heatmap

Attention is one of the major components of the proposed architecture. With attention, the decoder can scan through hidden vectors from the encoder and decide which part of the information to use when outputting a character by adaptively computing the attention weights. Attention maps along with the predicted texts for some example input images are given in Figure 11 and Figure 12.

Figure 9. Some samples of the augmented validation images

Figure 10. Some samples of the de-augmented validation images

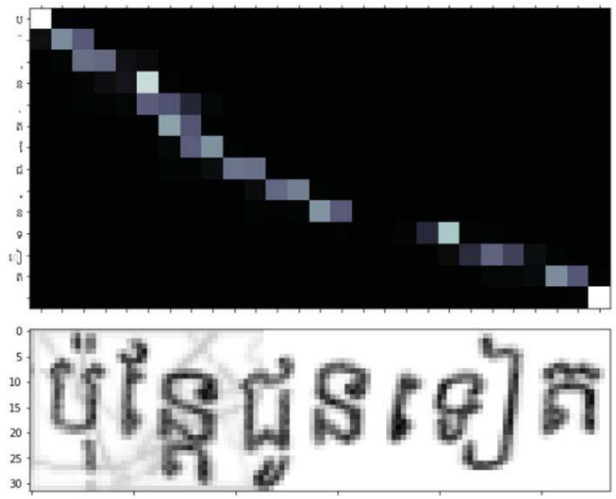

Figure 11. Attention maps along the predicted texts for some input images

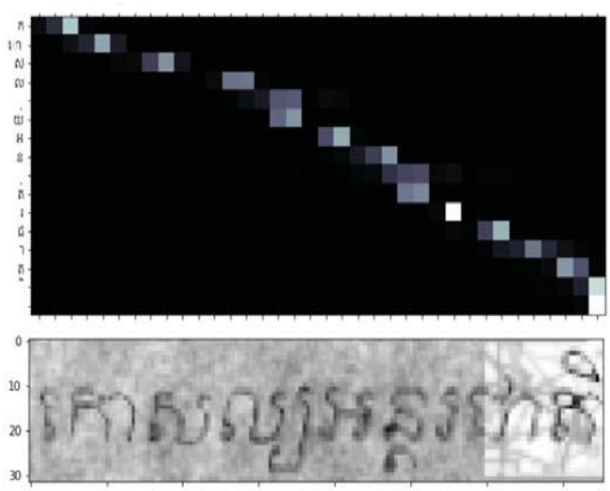

Figure 12. Attention maps along with the predicted texts for some input images

5. Discussion

The histograms of Levenshtein distances on the validation set for the trained model and Tesseract-OCR are given in Figure 13 and Figure 14, respectively. A few conclusions can be made:

• Maximum Levenshtein distance by Tesseract-OCR is three times higher than that of the proposed model;

• For Tesseract-OCR, the error is rather uniformly distributed between 10 and 60 (Levenshtein distance), while the for the proposed, the error distribution is right-skewed with most points located between 0 and 5;

• For Tesseract-OCR, the mean error is about 9.8, while for the proposed model, the mean error is about 0.2;

• The proposed model is more robust to complex data augmentation than Tesseract-OCR. The same observation was also made by Namysl and Konya (2019).

5.1. The model's performance in some extreme cases

To understand the model's performance better, we identified the images with the largest Levenshtein distances. A few identified images are illustrated in Figure 15 along with the predicted texts. Figure 15 shows that the recognition performance of the proposed model drops as text quality is significantly reduced due to the combined effects of augmentations. Some characters in Figure 15 are not clearly distinguishable even to human eyes.

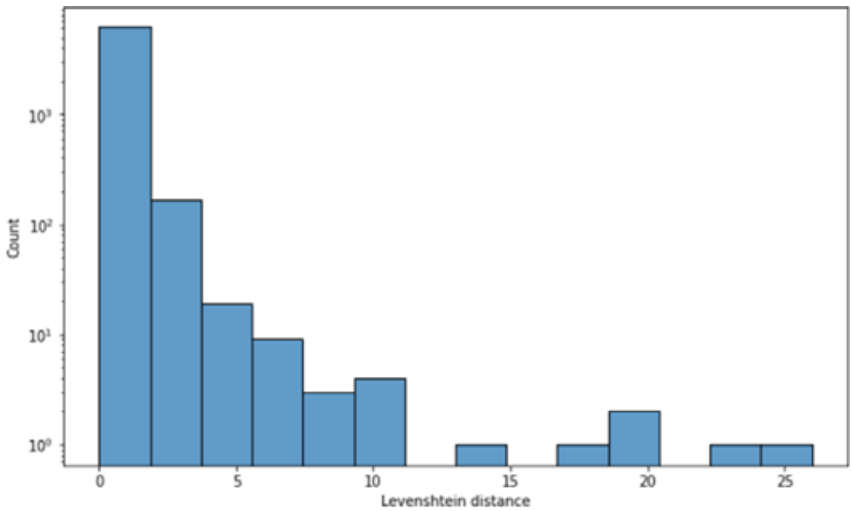

Figure 13. The model's histogram of Levenshtein distances on the validation set

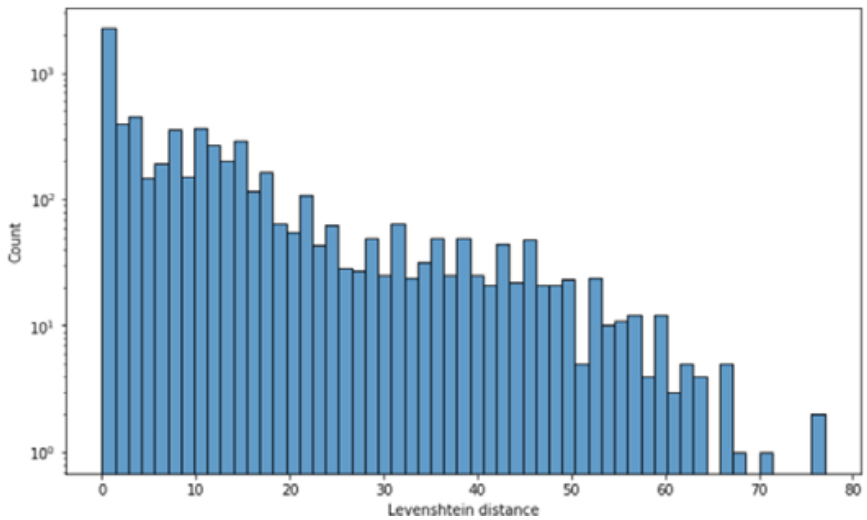

Figure 14. Tesseract-OCR's histogram of Levenshtein distances on the validation set

Figure 15. The proposed model's predictions on some extreme cases

6. Conclusion

This work presents an end-to-end Khmer OCR system that utilizes an encoder-decoder (Seq2Seq) network. The encoder is composed of layers of convolutional blocks and layers of GRU units. The decoder consists of a layer of GRU units and a linear classifier. The decoder uses an attention mechanism to adaptively select parts of the encoder's outputs that are relevant for predicting the target character. The proposed model has trained a collection of synthetic text-line images generated by using an open-source text2image tool with degrading effects. Data augmentation is further applied on both train and validation datasets to improve the model's robustness. The experiment results suggest that the proposed solution outperforms the current state-of-art Tesseract-OCR engine for Khmer by achieving a CER of 0.7% vs. 35.9% on the validation set of 6,400 augmented images. A robust OCR system must be able to recognize text in the presence of text deformations and noisy backgrounds. The system must also be font-invariant. In the future, we would like to train the model on more complex, augmented text-line images, and additional Khmer fonts are included.

References

Annanurov, B., & Noor, N. M. (2018). Khmer handwritten text recognition with convolution neural networks. *ARPN Journal of Engineering and Applied Sciences, 13*(22), 8828-8833.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate.* Retrieved October 10, 2021, from https://arxiv.org/pdf/1409.0473.pdf

Buoy, R., Taing, N., & Kor, S. (2020). *Khmer word segmentation using BiLSTM networks*. Paper presented at the 4th Regional Conference on OCR and NLP for ASEAN Languages (ONA 2020), Phnom Penh, Cambodia.

Buoy, R., Taing, N., & Kor, S. (2021). *Joint Khmer word segmentation and part-of-speech tagging using deep learning.* Retrieved October 10, 2021, from https://arxiv.org/ftp/arxiv/papers/2103/2103.16801.pdf

Chey, C., Kumhom, P., & Chamnongthai, K. (2005). Khmer printed character recognition by using wavelet descriptors. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 14*(3), 337-350.

Ding, C., Utiyama, M., & Sumita, E. (2018). NOVA: A feasible and flexible annotation system for joint tokenization and part-of-speech tagging. *ACM Transactions on Asian and Low-Resource Language Information Processing, 18*(2). doi:10.1145/3276773

Lenleng, I., & Muaz, A. (2015). Khmer Optical Character Recognition (OCR). *PAN Localization Cambodia, 1*.

Liebl, B., & Burghardt, M. (2020). *On the accuracy of CRNNs for line-based OCR: A multi-parameter evaluation.* Retrieved October 10, 2021, from https://arxiv.org/pdf/2008.02777.pdf

Memon, J., Sami, M., Khan, R. A., & Uddin, M. (2020). Handwritten Optical Character Recognition (OCR): A comprehensive Systematic Literature Review (SLR). *IEEE Access, 8*, 142642-142668.

Meng, H., & Morariu, D. (2014). *Khmer character recognition using artificial neural network.* Retrieved October 10, 2021, from http://www.apsipa.org/proceedings_2014/Data/paper/1408.pdf

Namysl, M., & Konya, I. (2019). *Efficient, lexicon-free OCR using deep learning.* Paper presented at the 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia.

Safir, F. B., Ohi, A. Q., Mridha, M. F., Monowar, M. M., & Hamid, M. A. (2021). *End-to-end optical character recognition for bengali handwritten words.* Retrieved October 10, 2021, from https://arxiv.org/pdf/2105.04020.pdf

Sahu, D. K., & Sukhwani, M. (2015). *Sequence to sequence learning for optical character recognition.* Retrieved October 10, 2021, from https://arxiv.org/pdf/1511.04176.pdf

Sok, M. (2016). *Phonological principles and automatic phonemic and phonetic transcription of khmer words.* Chiang Mai, Thailand: Payap University.

Sok, P., & Taing, N. (2014). *Support Vectir Machine(SVM)-based classifier for Khmer character-set recognition.* Retrieved October 10, 2021, from http://www.apsipa.org/proceedings_2014/data/paper/1407.pdf

Sokphyrum, K., Samak, S., & Sola, J. (2019). Khmer OCR finte tune engine for Unicode and legacy fonts using Tesseract 4.0 with Deep Neural Network. *Optical character recognition for complex scripts and Natural language processing for ASEAN Languages, 1*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks.* Retrieved October 10, 2021, from https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

Valy, D., Verleysen, M., Chhun, S., & Burie, J.-C. (2017). A new Khmer palm leaf manuscript dataset for document analysis and recognition: SleukRith set. *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing*, 1-6. doi:10.1145/3151509.3151510