applied sciences

MDPI

Article

Amharic OCR: An End-to-End Learning

Birhanu Belay [1,2,*], Tewodros Habtegebrial [1], Million Meshesha [3], Marcus Liwicki [4], Gebeyehu Belay [2] and Didier Stricker [1,5]

[1] Department of Computer Science, University of Kaiserslautern, 67653 Kaiserslautern, Germany; t_habtegeb15@cs.uni-kl.de (T.H.); didier.stricker@dfki.de (D.S.)
[2] Faculty of Computing, Bahir Dar Institute of Technology, Bahir Dar 6000, Ethiopia; ge.be09@yahoo.com
[3] School of Information studies for Africa, Addis Ababa University, Addis Ababa 1000, Ethiopia; meshe84@yahoo.com
[4] Department of Computer Science, Lulea University of Technology, 97187 Lulea, Sweden; marcus.liwicki@unifr.ch
[5] German Research Center for Artificial Intelligence, DFKI, 67663 Kaiserslautern, Germany
* Correspondence: b_belay18@cs.uni-kl.de

Received: 24 December 2019; Accepted: 30 January 2020; Published: 7 February 2020

Abstract: In this paper, we introduce an end-to-end Amharic text-line image recognition approach based on recurrent neural networks. Amharic is an indigenous Ethiopic script which follows a unique syllabic writing system adopted from an ancient Geez script. This script uses 34 consonant characters with the seven vowel variants of each (called basic characters) and other labialized characters derived by adding diacritical marks and/or removing parts of the basic characters. These associated diacritics on basic characters are relatively smaller in size, visually similar, and challenging to distinguish from the derived characters. Motivated by the recent success of end-to-end learning in pattern recognition, we propose a model which integrates a feature extractor, sequence learner, and transcriber in a unified module and then trained in an end-to-end fashion. The experimental results, on a printed and synthetic benchmark Amharic Optical Character Recognition (OCR) database called ADOCR, demonstrated that the proposed model outperforms state-of-the-art methods by 6.98% and 1.05%, respectively.

Keywords: Amharic script; CNN; CTC; end-to-end learning; LSTM; OCR; pattern recognition; text-line image

1. Introduction

Amharic is the second-largest Semitic dialect in the world after Arabic [1]. It is an official working language of the Federal Democratic Republic of Ethiopia and is spoken by more than 50 million people as their mother language and by over 100 million as a second language in the country [2,3]. In addition to Ethiopia, it is also spoken in other countries like Eritrea, USA, Israel, Sweden, Somalia, and Djibouti [1,4].

Dated back to the 12th century, many historical and literary documents in Ethiopia are written and documented using Amharic script. Amharic is a syllabic writing system which is derived from an ancient script called Geez, and it has been extensively used in all government and non-government sectors in Ethiopia until today. Amharic took all of the symbols in Geez and added some new ones that represent sounds not found in Geez [5].

In Amharic script, there are about 317 different alphabets including 238 core characters, 50 labialized characters, 9 punctuation marks, and 20 numerals which are written and read, as in English, from left to right [1,6]. All vowels and labialized characters in Amharic script are derived, with a small change, from the 34 consonant characters. The change involves modifying the structure of these characters by adding a straight line or shortening and/or elongating one of its main legs.

It also includes the addition of small diacritics, such as strokes or loops/circles to the right, left, top, or bottom of the character.

Due to these small modifications on the consonants, Amharic characters have similar shapes which may make the task of recognition hard for machines as well as humans [7]. These features are particularly interesting in research on character recognition because a small change in the basic physical features may affect the orthographic identities of letters. The shapes and structural formations of consonant Amharic characters with their corresponding vowels and labialized variants are depicted in Figure 1.

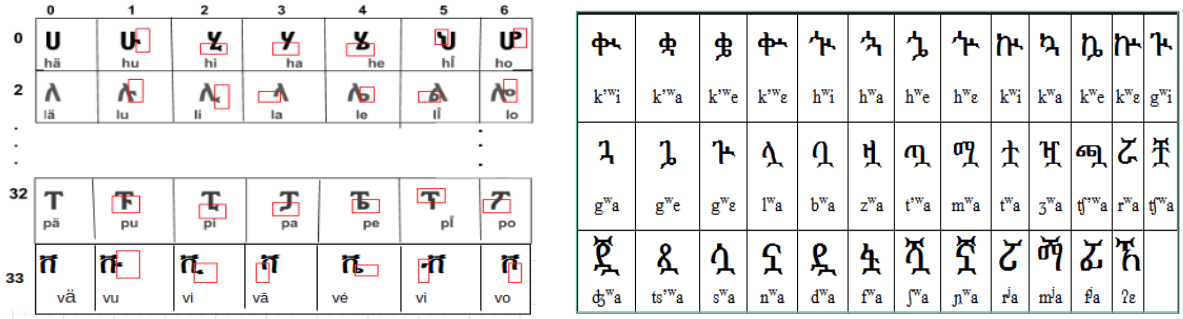

Figure 1. The shapes and structural formations of sample Amharic characters: (**a**) Basic Amharic characters with the orders of consonant–vowel variants (34 × 7), including the lately introduced Amharic character (ቨ). Characters in the first column are consonants and the others are derived variants (vowels) formed from each consonant by adding diacritics and/or removing part of the character, as marked with the red box. (**b**) Derived/labialized characters. Labialized characters marked with circles are also derived from consonant characters.

Optical Character Recognition (OCR) applications have been widely used and implemented for decades to digitize various historical and modern documents including books, newspapers, magazines, and cultural and religious archives that are written with different scripts. Multiple intensive works—for multiple scripts—have been done in the area of document image analysis with a better recognition accuracy; most of the scripts now even have commercial off-the-shelf OCR applications. In such a way, many researchers think that the OCR challenge is solved. However, OCR gives better results only for very specific use cases and there are still multiple indigenous scripts, like Amharic, which are underrepresented in the area of natural language processing (NLP) and document image analysis [8]. Until recent times, the OCR for Amharic script remained relatively unexplored, and it is still challenging [9,10].

Nowadays, Amharic document processing and preservation is given much attention by many researchers from the fields of computing, linguistics, and social science [3,11–13]. In recent years, various models and algorithms have been proposed by many researchers for image pattern recognition, and there has been a rapid advancement of solutions which have demonstrated ground-breaking performance [14].

The first work on Amharic OCR was done by Worku in 1997 [15]. Since then, attempts for Amharic OCR have also been made by employing different machine learning techniques. Here, we will try to cover the various techniques considered by different researchers. A tree classification scheme with the topological features of a character was used by Worku [15]. It was only able to recognize an Amharic character written with the Washera font and 12 point type.

Then, other attempts were made, such as typewritten [16], machine-printed [1], and Amharic braille document image recognition [17], Geez script written on vellum [18], Amharic document image recognition and retrieval [19], numeric recognition [20], and handwritten recognition [21]. However, all of these researchers applied statistical machine learning techniques and limited private datasets. In addition, recognition was done at the character level which is time consuming and also may not

achieve better recognition accuracy, since each image of a character is directly dependent on the nature of the segmentation algorithms [22].

Following the success of deep learning, other attempts have also been made for Amharic character recognition by employing Convolutional Neural Networks (CNNs) [6,8,23]. Nearly all of these attempts performed segmentation of text images into the character level, which also directly affects the performance of the OCR. The only exceptions are Assabie [21] and recently published works [9,24]. Assabie [21] proposed an Hidden Markov Model (HMM)-based model for offline handwritten Amharic word recognition without character segmentation by using the structural features of characters as building blocks of a recognition system, while Belay [9] and Addis [24] proposed Long–Short-Term Memory (LSTM) networks together with CTC (Connectionist Temporal Classification) for Amharic text image recognition. In literature, attempts at Amharic OCR have neither shown results on large datasets nor considered all possible characters used in the Amharic writing system [9].

There are many effective off-the-shelf commercial and open-source OCR applications for many languages, including the functionality of ground truth generation from the existing printed texts so as to train the second model [25]. The effectiveness of various open-source OCR engines was evaluated on 19th century Fraktur scripts; the evaluation shows that the open-source OCR engine can outperform the commercial OCR application [26].

Based on deep neural networks, many segmentation-based [27] and segmentation-free OCR techniques have been studied. Impressive progress has also been made for many Latin and non-Latin scripts, ranging from historical handwritten documents to modern machine-printed texts.

Bidirectional Recurrent Neural Networks with Long–Short-Term Memory (LSTM) architecture for online handwriting recognition [28], Convolutional Recurrent Neural Networks (CRNN) for Japanese handwriting recognition [29], Urdu Nastaleeq script recognition using bidirectional LSTM [30], segmentation-free Chinese handwritten text recognition [31], a hybrid Convolutional Long-Term Memory Network(CLSTM) for text image recognition [32], Multidimensional LSTM (MDLSTM) for Chinese handwriting recognition [33], combined Connectionist Temporal Classification (CTC) with Bidirectional LSTM (BLSTM) for unconstrained online handwriting recognition [34], MDLSTM for handwriting recognition [35], an online handwritten mathematical expression recognition using a Gated Recurrent Unit (GRU)-based attention mechanism [36], a combination of Convolutional Neural Networks and multi-dimensional RNN for Khmer historical handwritten text recognition [37], and a multi-stage HMM-based text recognition system for handwritten Arabic [38] have been studied.

However, OCR systems for many scripts, especially those which are indigenous to the African continent, such as Amharic, remain under-researched, and none of the researchers have taken advantage of deep learning techniques such as end-to-end learning, used for many languages, for developing Amharic OCR. Therefore, in this paper, we propose an end-to-end trainable neural network which includes a Convolutional Neural Network (CNN), Bidirectional LSTM (BLSTM), and Connectionist Temporal Classification (CTC) in a unified framework for Amharic text-line image recognition.

This paper is an extension of the previous work [9] with the following summarized contributions: (**1**) To extract automatic features from text-line images, we propose a CNN-based feature extractor module. (**2**) To reduce the computational cost, we adjust the images to a smaller size. (**3**) We adopt an end-to-end trainable neural network for Amharic text-line image recognition that achieves state-of-the-art results. (**4**) We also use an extra private Amharic dataset to tune the parameters of the feature extractor module. (**5**) Based on the experimental results obtained, a detailed analysis of the dataset is presented.

2. Material and Methods

A general OCR application starts from dataset preparation and then followed by model training. In this section, we will present the dataset used for training and model evaluation, the proposed model architecture, and the training schemes.

2.1. Datasets

The shortage of datasets is the main challenge in pattern recognition, and it is one of the limiting factors in developing reliable systems for Amharic OCR. There are few databases used in the various works on Amharic OCR reported in literature. As reported in [16], the authors considered the most frequently used Amharic characters. A later work by Million et al. [1] uses 76,800 character images with different font types and sizes which belong to 231 classes.

Other researchers' work on Amharic OCR [15,20,21] reported that they used their own private databases, but none of them were made publicly available for research purposes. Promising work on Amharic OCR is reported in [9], where the authors employed an LSTM-based neural network for Amharic OCR text-line image recognition; they also introduce a dataset called ADOCR, which is made public and freely accessible at http://www.dfki.uni-kl.de/~belay/.

In this paper, we use the ADOCR database introduced by [9]. This dataset contains 337,337 Amharic text-line images which are written with the Visual Geez and Power Geez fonts using 280 unique Amharic characters and punctuation marks.All images are greyscale and normalized to 48 by 128 pixels, while the maximum string length of the ground-truth text is 32 characters including true blank spaces.

From the total text-line images in the dataset, 40,929 are printed text-line images written with the Power Geez font; 197,484 and 98,924 images are synthetic text-line images generated with different levels of degradation using the Power Geez and Visual Geez fonts, respectively. All characters that exist in the test dataset also exist in the training set, but some words in the test dataset do not exist in training dataset. Sample printed and synthetically generated text-line images taken from the database are shown in Figure 2.

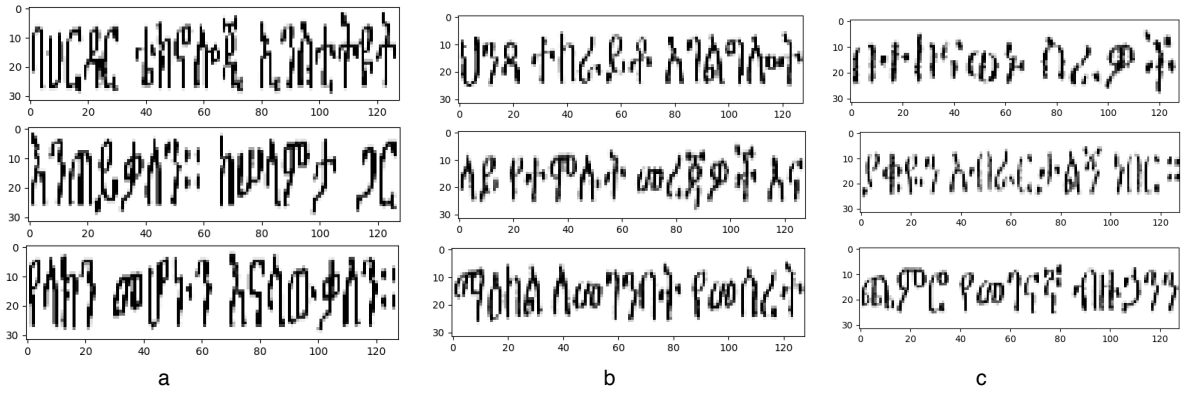

Figure 2. Sample Amharic text-line images that are normalized to a size of 32 by 128 pixels: (**a**) Printed text-line images written with the Power Geez font type. (**b**) Synthetically generated text-line images with the Visual Geez font type. (**c**) Synthetically generated text-line images with the Power Geez font type.

The details of the dataset (text-line images, unique Amharic characters, and punctuation marks) used in this experiment are summarized in Table 1 and Figure 3.

Table 1. The details of the text-line images in the dataset [9].

	Printed	**Synthetic**	
Font Type	**Power Geez**	**Power Geez**	**Visual Geez**
Number of samples	40,929	197,484	98,924
No. of test samples	2907	9245	6479
No. of training samples	38,022	188,239	92,445
No. of unique chars.	280	261	210

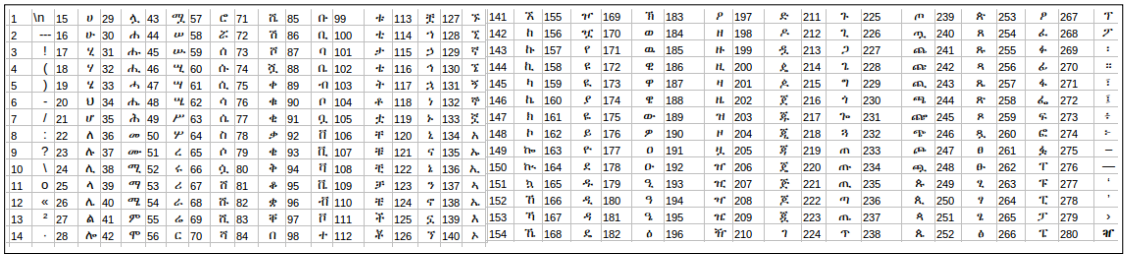

Figure 3. Unique Amharic characters and punctuation marks presented in the ground truth of the Amharic OCR database [9].

2.2. Proposed Model

For the sequence recognition problem, the most suited neural networks are recurrent neural networks (RNN) [39], while for an image-based problem, the most suited are convolutional neural networks (CNN) [40]. To work with the OCR problems, it would be better to combine CNNs and RNNs. The overall framework of the proposed approach is shown in Figure 4. In this framework, we employed three modules: The feature extractor, the sequence learner, and the transcriber module. All three of these modules are integrated in a single framework and trained in an end-to-end fashion.

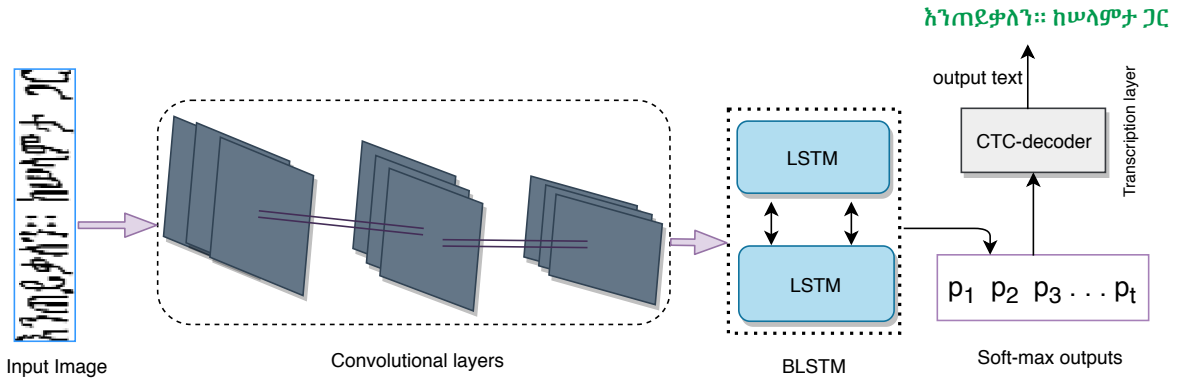

Figure 4. The proposed model. This network consists of three components. The first component is the convolutional layer which acts as the feature extractor, the second component is the recurrent layer which acts as a sequence learner, plus a soft-max function to predict a label distribution at each time-step. The third component is a Connectionist Temporal Classification (CTC) layer which takes the soft-max prediction at each time-step as an input and transcribes these predictions into final label sequences. All input text-line images are normalized to a fixed size of 32 by 128 pixels, and labels are padded with zero until they reach the maximum sequence length of 32.

The feature extractor module consists of seven convolutional layers which have a kernel size of 3×3, except the one that is stacked on top with a 2×2 kernel size; each uses rectifier linear unit (ReLU) activation, four max-pooling layers with pool sizes of 2 for the first pooling layer, and 2×1 for the remaining pooling layers. Strides are fixed to one, and the 'same' padding is used in all convectional layers. The number of feature maps is gradually increased from 64 to 512, while the image size is quickly reduced by a spatial pooling to further increase the depth of the network. For an input size N, kernel K, padding P, and stride S, the output size M at each convolutional layer can be computed as $M = (\frac{N-K+2*P}{S}) + 1$. In addition, to normalize the output of the previous activation layer and accelerate the training process, batch normalization is used after the fifth and sixth convolutional layers.

We also used a reshape function to make the output of the convolutional layer compatible with the LSTM layer. The sequence learner module is the middle layer of our framework which predicts the sequential output per time-step. This module consists of two bidirectional LSTM layers, with the soft-max function on top, each of which has 128 hidden layer sizes and a dropout rate of 0.25. The sequential output of each time-step from the LSTM layers is fed into a soft-max layer to

get a probability distribution over the C possible characters. Finally, transcription to the equivalent characters is done using the CTC layer. The details of the network parameters and configuration of the proposed model are depicted in Tables 2 and 3.

Table 2. Convolutional network layers of the proposed model and their corresponding parameter values for an input image size $32 \times 128 \times 1$.

Network Layers	Kernel Size	Stride	Feature Maps
Convolution	3×3	1×1	64
Max-Pooling	2×2	2×2	-
Convolution	3×3	1×1	128
Max-Pooling	2×1	1×1	-
Convolution	3×3	1×1	256
BatchNormalization	-	-	-
Convolution	3×3	1×1	256
BatchNormalization	-	-	-
Max-Pooling	2×1	1×1	-
Convolution	3×3	1×1	512

Table 3. The recurrent network layers of the proposed model with their corresponding parameter values. The input size of the Long–Short-Term Memory (LSTM) is a squeezed output of the convolutional layers, which is depicted in Table 2.

Network Layers (Type)	Hidden Layer Size
BLSTM	128
BLSTM	128
Soft-Max	No. class = 281

During training, the image passes through the convolutional layers, in which several filters extract features from the input images. After passing some convolutional layers in sequence, the output is reshaped and connected to a bidirectional LSTM. The output of the LSTM is fed into a soft-max function which has $n + 1$ nodes, where each node corresponds to a label or each unique character in the ground truth, including one blank character which is used to take care of the continuous occurrence of the same characters. In our case, since there are 280 unique characters, the soft-max outputs will be 281 probabilities, including the blank character, at each time-step. We employed a checkpoint strategy which can save the model weight to the same file each time an improvement is observed in validation loss.

As explained in the work of Graves et al. [41], CTC is an objective function which adopts dynamic programming algorithms to directly learn the alignment between the input and output sequences. Then, the CTC loss function is used to train the network. During training, CTC only requires an input sequence and the corresponding transcriptions. For given training data D, CTC minimizes the negative logarithm of the likelihood loss function, formulated as Equation (1).

$$l_{CTC} = -log\left(\prod_{(x,z)\in D} p(z/x)\right), \tag{1}$$

where $x = x_1, x_2, \ldots, x_T$ is the input sequence with length T, $z = z_1, z_2, \ldots, z_C$ is the corresponding target for $C < T$,and the p(z/x) is computed by multiplying the probability of labels along the path π that contains the output label over all time-steps t, as shown in Equation (2).

$$p(\pi/x) = \prod_{t}^{T} p(\pi_t, t/x), \tag{2}$$

where t is the time-step and π_t is the label of path π at t.

A target label in path π is obtained by mapping the reduction function B that converts a sequence of soft-max outputs for each frame into a label sequence by removing repeated labels and blank (ϕ) tokens of the given sequences. Taking an example from [9], for a given sequence of observation (o) with a length of eighteen, $o = \phi aa\phi mm\phi h\phi aa\phi rr\phi ic\phi$. Then, the paths are mapped to a label sequence $l_s = B(o) = B(\phi aa\phi mm\phi h\phi aa\phi rr\phi ic\phi) = B(\phi a\phi m\phi h\phi a\phi r\phi ic\phi)$ ='amharic', where B is a reduction mapping function which works by first collapsing the repeated tokens and then removing blanks.

The target sequence probability y from input sequence x is the sum of the probability of all paths by reducing each path to this label sequence using B, and it is formulated as Equation (3).

$$p(y/x) = \sum_{\pi \in B(y)} p(\pi/x) \tag{3}$$

Once the probability of label sequence y from an input sequence x is obtained with the CTC forward–backward algorithm proposed by [41], we employ the best path decoding method, fast and simple, to find a character (C) that has the highest score from outputs (i) at every time-step; the final recognized string text can be generated using B without segmentation of the input sequence; this is formulated as Equation (4).

$$C_{\max} = B(arg\,max_i(y_i^t)), for\ t = 1, 2...T \tag{4}$$

In all experiments and results reported below, we used the same network architecture, and the performance of the proposed model is described in terms of Character Error Rate (CER), which is computed by counting the number of characters inserted, substituted, and deleted in each sequence and then dividing by the total number of characters in the ground truth; this can be formulated as Equation (5).

$$CER(P, T) = \left(\frac{1}{q} \sum_{n \in P, m \in T} D(n, m) \right) \times 100, \tag{5}$$

where q is the total number of target character labels in the ground truth, P and T are the predicted and ground-truth labels, and $D(n, m)$ is the edit distance between sequences n and m.

3. Experimental Results

Experiments were conducted using the ADOCR database [9], a public and freely available dataset, which contains both printed and synthetically generated Amharic text-line images. Following the network architecture and experimental setups described in Section 2, we implemented our model with the Keras Application Program Interface (API) with a TensorFlow backend, and the model was trained on a GeForce GTX 1070 GPU.

To select suitable network parameters, different values of these parameters were considered and tuned during experimentation, and the results reported in this paper were obtained using an Adam optimizer employing a convolutional neural network with a feature map that started from 64 and increased to 512, the BLSTM network with two network hidden layers with sizes of 128 each, and a learning rate of 0.001.

Based on the nature of the dataset, we conducted three experiments. Once we trained our network with the synthetic and some of the printed text-line images, the performance of the model was tested with three different test datasets. In the first and the second experiments, the model was evaluated with synthetic Amharic text-line images generated with the Power Geez and Visual Geez fonts, respectively. The third experiment was conducted using a printed test dataset written with the Power Geez font type.

In the original dataset, the sizes of the images were 48 by 128 pixels. Considering similar works done in the area, to reduce computational costs during training, we resized the images into sizes of 32

by 128 pixels. For validation, we used 7% of the training dataset, randomly selected, as proposed in the original paper [9]. The network was trained for 10 epochs with a batch size of 200.

During the testing of the proposed model, character error rates of 1.05% and 3.73% were recorded on the two test datasets which were generated synthetically using the Visual Geez and Power Geez fonts, respectively. The model was also tested with the third test dataset, which had printed text-line images that were written with the Power Geez fonts, and a character error rate of 1.59% was obtained. The results recorded during experimentation are summarized in Table 4.

Table 4. Experimental results (Character Error Rate (CER)).

Image Type	Font Type	Test Data Size	CER (%)
Printed	Power Geez	2907	1.59
Synthetic	Visual Geez	6479	1.05
	Power Geez	9245	3.73

4. Discussion and Analysis of the Results

We performed repetitive evaluations of our method using the benchmark datasets used in the original paper [9], and we also tried to compare with the state-of-the-art methods on both printed and synthetically generated datasets. The details of the analysis and comparisons are presented in the following sections.

4.1. Analysis of the Dataset and Results

In this section, a detailed analysis and description of the dataset are presented, and then the results obtained during experimentation will be presented. As depicted in Figure 5, we observed that some of the printed text-line images are not properly aligned with the ground truth due to the occurrence of extra blank spaces between words and/or the merging together of more words during printing. In addition, with the synthetic text-line images, a character at the beginning and/or at the end of the word is missed, which results in misalignment with the ground-truth. To improve the recognition performance, it is important to annotate the data manually or to use better data annotation tools.

Of several factors, samples with wrongly annotated Amharic text-line images and characters, depicted in Figure 5, are the majors factors causing recognition errors. In general, the recognition errors may occur due to misspelled characters, spurious symbols, or lost characters.

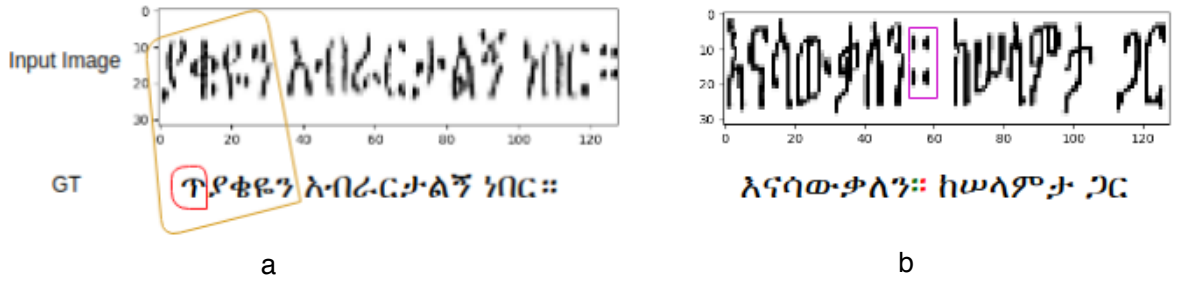

Figure 5. Samples with wrongly annotated images and GT from the test dataset. (**a**) Synthetic text-line image; the word marked with yellow rectangle is a sample mislabeled word where the first character (**ጥ**) in the GT, marked with a red circle, is missed in the input image but it exists in GT. (**b**) Printed text-line image; a punctuation mark called a full stop/period, bounded with a purple rectangle in the input image, is incorrectly labeled as two other punctuation marks called word separators, indicated by the green and red text colors in the GT.

The proposed model works better on the synthetic test datasets generated with the Visual Geez font type, compared to the character error rate observed on the Power Geez font type and the printed test data. The generalization of this model, especially on synthetic dataset generated with the Power

Geez font type, is not as good as the Visual Geez one. This happens mainly because of the significantly larger number of text-line images in the test set and the nature of the training samples (i.e., the text-line images and the ground truth are not properly annotated due to the existence of deformed characters and missing characters in the beginning and/or end of the text-line images during data generation but not in the ground truth). In addition, text-line images generated with the Power Geez font are relatively blurred, resulting in poor recognition accuracy.

In Figure 6, sample Amharic text-line images and predicted texts containing different types of character errors during testing are shown.

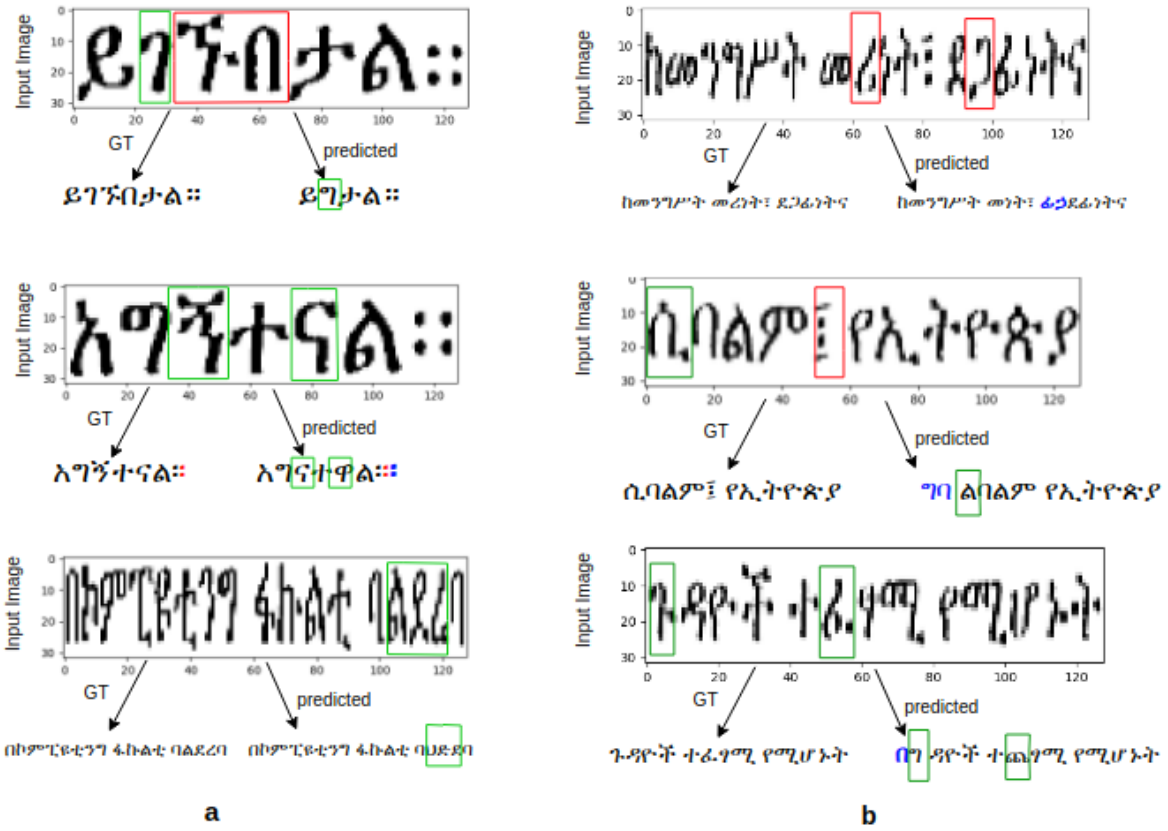

Figure 6. Sample mis-recognized text-line images. (**a**) Printed text-line images. (**b**) Synthetically generated text-line images. The small green and red rectangles are used to mark the substituted and deleted characters, respectively, while the blue color represents inserted characters.

For example, as illustrated at the top of Figure 5a, the character (ነ) marked by the green rectangle in the input image is substituted by the character (ግ) in the predicted text. On the other hand, the two characters (ጉ and በ) marked with the red rectangle are deleted characters. Other character errors are depicted at the top of Figure 6b; characters (ሪ and ጋ) from the input image, marked with the red rectangles, are deleted characters, while characters (ፈ and ኃ) in the predicted text, written with a blue color, are inserted characters.

4.2. Performance Comparison

As depicted in Figure 7 and Table 5, the performance of the proposed model is improved. Compared to the original paper, the proposed model achieved better recognition performance with a smaller number of epochs. However, the proposed model took a longer time for training. This is due to the nature of an end-to-end learning approach [42] that incorporates multiple and diverse network layers in a single unified framework. Therefore, the proposed model can be assessed and

training time may be further improved by following some other concepts like decomposition [43] or the divide-and-conquer approach.

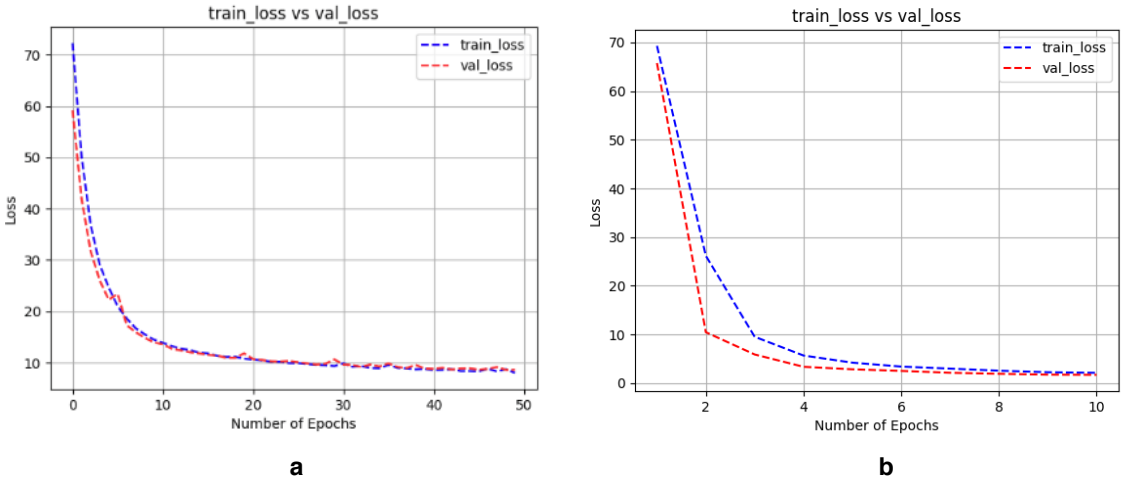

Figure 7. Learning loss comparison: (**a**) The training and validation losses of CTC in the original paper [9] recorded for 50 epochs. (**b**) The training and validation CTC-loss of the proposed model recorded for 10 epochs.

The comparisons among the proposed approach and others' attempts done on the ADOCR Database [9] are listen in Table 5, and the performance of the proposed model shows better results on all three of the test datasets.

Table 5. Comparison of test results (CER).

	No. of Text-Lines	Image Type	Font Type	CER (%)
Addis [24] *	12 pages	printed	-	2.12%
Belay [9]	2,907	Printed	Power Geez	8.54%
Belay [9]	9,245	Synthetic	Power Geez	4.24%
Belay [9]	6,479	Synthetic	Visual Geez	2.28%
Ours	2,907	Printed	Power Geez	1.56%
Ours	9,245	Synthetic	Power Geez	3.73%
Ours	6,479	Synthetic	Visual Geez	1.05%

* Denotes methods tested on different datasets.

All results are reported as character error rates (CERs) as results of insertion, deletion, and substitution of characters in the predicted output. This can be efficiently computed using Equation (5).

5. Conclusions

In this paper, we propose a method for text-line image recognition of Amharic, an old Semitic language. Amharic has it own indigenous script and is rich in a bulk of historically printed and handwritten documents. However, it is an underprivileged group of scripts in Natural Language Processing (NLP) due to the lack of extensive research in the area and the lack of annotated datasets. Therefore, in this paper, we present an end-to-end trainable neural network architecture which consists of CNN (the feature extractor), LSTM (the predictor), and CTC (the transcriber) in a unified framework. The proposed model is evaluated using a publicly available Amharic database called ADOCR, and it outperforms the state-of-art methods employed on this benchmark dataset [9] by a large margin.

As part of future work, the proposed method will extended for handwritten Amharic document image recognition. Similarly, we have planned to develop an OCR system that should recognize some complex Amharic documents, such as historical and scene Amharic text images.

Author Contributions: Conceptualization, B.B.; Methodology, B.B. and T.H.; Validation, B.B.; Writing—original draft preparation, B.B.; Writing—review and editing, T.H., M.M., and G.B. Supervision, M.L. and D.S. All authors have read and agreed to the published version of the manuscript.

Funding: The first author was supported by a DAAD scholarship (Funding program No. 57375975). This research was carried out at the Augmented Vision lab at DFKI, Kaiserslautern, Germany, and it was partially funded by the BMBF project VIDETE (01IW1800).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Meshesha, M.; Jawahar, C. Optical character recognition of amharic documents. *Afr. J. Inf. Commun. Technol.* **2007**, *3*. doi:10.5130/ajict.v3i2.543.
2. The Ethiopian Press Agency. Will Amharic be AU's Lingua Franca ? *The Ethiopian Herald*. February 2019. Available online: https://www.press.et/english/?p=2654#l (accessed on 10 November 2019).
3. Getahun, M. Amharic Text Document Summarization using Parser. *Int. J. Pure Appl. Math.* **2018**, *118*, 24.
4. Meyer, R. Amharic as lingua franca in ethiopia. *Lissan J. Afr. Lang. Linguist.* **2006**, *20*, 117–132.
5. Atelach, A.; Lars, A.; Mesfin, G. Natural Language Processing for Amharic: Overview and Suggestions for a Way Forward. In Proceedings of the 10th Conference on Traitement Automatique des Langues Naturelles, Batzsur-Mer, France, 11–14 June 2003; Volume 2, pp. 173–182.
6. Belay, B.; Habtegebrial, T.; Liwicki, M.; Belay, G.; Stricker, D. Factored Convolutional Neural Network for Amharic Character Image Recognition. In the Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 2906–2910.
7. Bloor, T. The Ethiopic Writing System: A Profile. *J. Simpl. Spell. Soc.* **1995**, *19*, 30–36,
8. Belay, B.; Habtegebrial, T.; Stricker, D. Amharic character image recognition. In Proceedings of the IEEE International Conference on Communication Technology (ICCT), Chongqing, China, 8–11 October 2018; pp. 1179–1182
9. Belay, B.; Habtegebrial, T.; Liwicki, M.; Belay, G.; Stricker, D. Amharic Text Image Recognition: Dataset, Algorithm and Analysis. In Proceedings of the IEEE International Conference Document Analysis and Recognition (ICDAR), Sydney, Australia, 20–25 September 2019; pp. 1268–1273
10. Weldegebriel, H.; Chen, J.; Zhang, D. Deep learning for Ethiopian Ge'ez script optical character recognision. In Proceedings of the 2008 Tenth International Conference on Advanced Computational Intelligence (ICACI), Xiamen, China, 29–31 March 2018; pp. 540–545.
11. Nirayo, H.; Andreas, N. An Amharic Syllable-Based Speech Corpus for Continuous Speech Recognition. In *International Conference on Statistical Language and Speech Processing*; Springer: Cham, Switzerland, 2019; pp. 177–187.
12. Abate, S.T.; Melese, M.; Tachbelie, M.Y.; Meshesha, M.; Atinafu, S.; Mulugeta, W.; Assabie, Y.; Abera, H.; Seyoum, B.E.; Abebe, T.; et al. Parallel Corpora for bi-lingual English-Ethiopian Languages Statistical Machine Translation. In Proceedings of the International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 3102–3111.
13. Bulakh, M.; Hummel, S.; Panini, F. Bibliography of Ethiopian Semitic, Cushitic and Omotic Linguistics XXI: 2017. *Int. J. Ethiop. Eritrean Stud. Aethiop.* **2018**, *Aethiopica 21*, 217–225.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
15. Worku, A. The Application of Ocr Techniques to the Amharic Script. Master's Thesis, Addis Ababa University Faculty of Informatics, Addis Ababa, Ethiopia, 1997.
16. Dereje, T. Optical Character Recognition of Type-Written Amharic Text. Master's Thesis, School of Information Studies for Africa, Addis Ababa, Ethiopia, 1999.
17. Hassen, S.; Yaregal, A. Recognition of double sided amharic braille documents. *Int. J. Image. Graph. Signal Process.* **2017**, *9*, 1.
18. Tegen, S. Optical Character Recognition for GE'Ez Scripts Written on the Vellum. Ph.D. Thesis, University of Gondar, Gondar, Ethiopia, 2017.
19. Million, M. Recognition and Retrieval from Document Image Collections. Ph.D. Thesis, IIIT Hyderabad, Hyderabad, India, 2008.

20. Betselot, R.; Dhara, R.; Gayatri V. Amharic handwritten character recognition using combined features and support vector machine. In Proceedings of the International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 11–12 May 2018; pp. 265–270.
21. Yaregal, A.; Josef, B. Hmm-based handwritten amharic word recognition with feature concatenation. In Proceedings of the International Conference on Document Analysis and Recognition, Barcelona, Spain, 26–29 July 2009; pp. 961–965
22. Din, I.; Siddiqi, I.; Khalid, S.; Azam, T. Segmentation-free optical character recognition for printed Urdu text. *EURASIP J. Image Video Process.* **2017**, *62*. doi:10.1186/s13640-017-0208-z.
23. Messay, S.; Schmidt, L.; Boltena, A.; Jomaa, S. Handwritten Amharic Character Recognition Using a Convolutional Neural Network. *arXiv* **2019**, arXiv:1909.12943.
24. Addis, D.; Liu, C.; Ta, D. Printed Ethiopic Script Recognition by Using LSTM Networks. In Proceedings of the International Conference on System Science and Engineering (ICSSE), Taipei, Taiwan, 28–30 June 2018; pp. 1–6.
25. Rigaud, C.; Burie, J.; Ogier, M. Segmentation-free speech text recognition for comic books. In Proceeding of the International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 3, pp. 29–34,
26. Reul, C.; Springmann, U.; Wick, C.; Puppe, F. State of the art optical character recognition of 19th century fraktur scripts using open source engines. *arXiv* **2018**, arXiv:1810.03436.
27. Husnain, M.; Saad Missen, M.; Mumtaz, S.; Jhanidr, M.; Coustaty, M.; Muzzamil, M.; Ogier, J.; Sang, G. Recognition of Urdu Handwritten Characters Using Convolutional Neural Network. *Appl. Sci.* **2019**, *9*, 2758.
28. Liwicki, M.; Graves, A.; Fern'andez, S.; Bunke, H.; Schmidhuber, J. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007, Curitiba, Brazil, 23–26 September 2007.
29. Ly, T.; Nguyen, C.; Nguyen, K.; Nakagawa, M. Deep convolutional recurrent network for segmentation-free offline handwritten Japanese text recognition. In Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 7, pp. 5–9.
30. Ul-Hasan, A.; Ahmed, S.B.; Rashid, F.; Shafait, F.; Breuel, T.M. Offline printed Urdu Nastaleeq script recognition with bidirectional LSTM networks. In Proceedings of the 2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, 25–28 August 2013; pp. 1061–1065.
31. Messina, R.; Louradour, J. Segmentation-free handwritten Chinese text recognition with lstm-rnn. In Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR), Tunis, Tunisia, 23–26 August 2015; pp. 171–175.
32. Breuel, T. High performance text recognition using a hybrid convolutional-lstm implementation. In Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 1, pp. 11–16.
33. Wu, Y.; Yin, F. Chen, Z.; Liu, L. Handwritten Chinese text recognition using separable multi-dimensional recurrent neural network. In Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 1, pp. 79–84.
34. Graves, A.; Liwicki, M.; Bunke, H.; Schmidhuber, J.; Fernandz, S. Unconstrained on-line handwriting recognition with recurrent neural networks. In Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–11 December 2008; pp. 577–584.
35. Castro, D.; Bezerra, B.L.; Valença, M. Boosting the deep multidimensional long-short-term memory network for handwritten recognition systems. In Proceedings of the 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, NY, USA, 5–8 August 2018; pp. 127–132.
36. Zhang, J.; Du, J.; Dai, L. A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition. In Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017 ; Volume 1, pp. 902–907.
37. Valy, D.; Verleysen, M.; Chhun, S. Text Recognition on Khmer Historical Documents using Glyph Class Map Generation with Encoder-Decoder Model. In Proceedings of the ICPRAM, Prague, Czech Republic, 19–21 February 2019.
38. Ahmad, I.; Fink, G. Handwritten Arabic text recognition using multi-stage sub-core-shape HMMs. *Int. J. Doc. Anal. Recognit. (IJDAR)* **2019**, *22*, 329–349.

39. Ghosh, R.; Vamshi, C.; Kumar, P. RNN based online handwritten word recognition in Devanagari and Bengali scripts using horizontal zoning. *Pattern Recognit.* **2019**, *92*, 203–218.
40. Hijazi, S.; Kumar, R.; Rowen, C. *Using Convolutional Neural Networks for Image Recognition*; Cadence Design Systems Inc.: San Jose, CA, USA, 2015.
41. Graves, A.; Fernandez, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 369–376.
42. Glasmachers, T. Limits of end-to-end learning. *arXiv* **2017**, arXiv:1704.08305.
43. Shalev-Shwartz, S.; Shamir, O.; Shammah, S. Failures of gradient-based deep learning. In Proceedings of the 34th International Conference on Machine Learning- Sydney, Australia, 6–11 August 2017; Volume 70, pp. 3067–3075.