Research Article

AHWR-Net: offline handwritten amharic word recognition using convolutional recurrent neural network

Fetulhak Abdurahman¹ · Eyob Sisay¹ · Kinde Anlay Fante¹

Received: 15 April 2021 / Accepted: 19 July 2021

Published online: 30 July 2021

© The Author(s) 2021 [OPEN](#)

Abstract

Amharic (አማርኛ) is the official language of the Federal Government of Ethiopia, with more than 27 million speakers. It uses an Ethiopic script, which has 238 core and 27 labialized characters. It is a low-resourced language, and a few attempts have been made so far for its handwritten text recognition. However, Amharic handwritten text recognition is challenging due to the very high similarity between characters. This paper presents a convolutional recurrent neural networks based offline handwritten Amharic word recognition system. The proposed framework comprises convolutional neural networks (CNNs) for feature extraction from input word images, recurrent neural network (RNNs) for sequence encoding, and connectionist temporal classification as a loss function. We designed a custom CNN model and compared its performance with three different state-of-the-art CNN models, including DenseNet-121, ResNet-50 and VGG-19 after modifying their architectures to fit our problem domain, for robust feature extraction from handwritten Amharic word images. We have conducted detailed experiments with different CNN and RNN architectures, input word image sizes, and applied data augmentation techniques to enhance performance of the proposed models. We have prepared a handwritten Amharic word dataset, HARD-I, which is available publicly for researchers. From the experiments on various recognition models using our dataset, a WER of 5.24 % and CER of 1.15 % were achieved using our best-performing recognition model. The proposed models achieve a competitive performance compared to existing models for offline handwritten Amharic word recognition.

Keywords Amharic · CNN · CTC · Handwritten · LSTM · Recognition

1 Introduction

The classic Amharic script was tailored to classical languages such as Ge'ez, and so many new signs have been derived for modern Amharic language. As a subgroup within these, Amharic (አማርኛ) is one of the Ethiopian Semitic languages which uses Ethiopic script. It is the official language of the Federal Government of Ethiopia which has more than 27 million speakers.

Amharic language has 34 base characters from which six or more than six (if the base character has labialized

forms) families are formed by changing the shape of the base characters (e.g. ለሉሊ ላላልሎሊ). There is a similarity in shape between inter-family or intra-family characters, which differs with a single stroke. There are a total of 238 core characters (34 base characters with six orders representing derived vocal sounds of the base character) and 27 labialized characters (which have two sounds such as ሊ ሊ፡ ሊ፡ ሊ፡) [1]. Amharic language alphabet, known as "Fidel" is written in tabular form with seven columns where the first column represents base characters while the remaining six or more columns represent derived characters, as

✉ Fetulhak Abdurahman, afetulhak@yahoo.com; Eyob Sisay, eysilka@gmail.com; Kinde Anlay Fante, kinde.anlay@ju.edu.et | ¹Faculty of Electrical and Computer Engineering, JiT, Jimma University, Jimma, Ethiopia.

(a)

Base Sound		Orders						
		1 st (ä)	2 nd (u)	3 rd (i)	4 th (a)	5 th (e)	6 th (ä)	7 th (o)
1	h	ሀ	ሀ	ሀ	ሀ	ሀ	ሀ	ሀ
2	l	ለ	ለ	ለ	ለ	ለ	ለ	ለ
3	h	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ
4	m	መ	መ	መ	መ	መ	መ	መ
5	s	ሠ	ሠ	ሠ	ሠ	ሠ	ሠ	ሠ
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
32	f	ፈ	ፈ	ፈ	ፈ	ፈ	ፈ	ፈ
33	p	ፑ	ፑ	ፑ	ፑ	ፑ	ፑ	ፑ
34	v	ቨ	ቨ	ቨ	ቨ	ቨ	ቨ	ቨ

(b)

Fig. 1 **a** A sample handwritten Amharic text. **b** A sample Amharic characters in tabular form having 7 columns. The red circles in the first row indicate how the derived characters differ from the base character in column 1 with a single stroke.

shown in Fig. 1b. Among the base characters, some of them do not have a unique sound and can be used interchangeably in written documents (such copairs are shown in braces: ሀ, ሐ, ሐ, ሐ, ሐ). Amharic text is written like the Latin script from left to right, and words are separated by blank space and top to bottom for newlines. Unlike the Latin script, there are no lowercase and uppercase letters. A sample handwritten text is shown in Fig. 1a.

Texts can be generated either in printed or handwritten form, and they have variability, which comes from various printing fonts or writing style of different individuals. Handwritten text recognition (HTR) system aims to transform images of handwritten text into editable text. Despite the remarkable development of digital text processing technology, handwriting is still used in our day-to-day life. The problem with printed and handwritten documents is its difficulty in sharing, storing, and efficiently managing the document. The conversion of handwritten documents has a great significance in various application areas, including preserving historical heritage by converting historical documents, bank check processing, postal address sorting, and many more.

HTR can be designed either for online or offline recognition. Concisely, online HTR is done at the time of writing where temporal features can be captured from both pen trajectory and the resulting image, whereas offline HTR is done only from a scanned image of a handwritten document after writing is completed. HTR is a challenging task due to the cursive nature of hand-writings, the complicated background of text images, myriad different writing styles (the calligraphy) and other language-related issues. Specifically, the high similarity in shape between

inter-family or intra-family characters in Amharic language makes the HTR task extremely challenging.

Researchers have been working for several decades on the development of HTR systems. In traditional HTR approaches, different image processing techniques were applied for segmenting the handwritten document into lines, words or characters. After segmentation, hand-engineered features were extracted and hidden Markov models (HMMs) [2, 3] were applied as sequence learning algorithms to represent the output as a character sequence. However, HMMs have limitation due to the markovian assumption, which learns text context information only from the current state, making it challenging to model contextual effects.

The advancement of deep learning algorithms has led us to use them for handwritten recognition tasks. Recurrent neural networks (RNNs) and multidimensional RNNs (MDRNNs) can overcome the limitations of traditional HMM-based sequence learning models. Despite their ability to handle the markovian assumptions, conventional RNNs require segmented input at each time step due to the behavior of their loss function. Researchers attempted to solve this problem using a hybrid model by integrating HMMs with RNNs [4]. The HMMs in hybrid models inhibit the usage of RNNs to their full potential. The introduction of connectionist temporal classification (CTC) algorithm [5] eliminates the need for segmented inputs in traditional RNN based models by decoding output sequences without a one-to-one correspondence between input sequences. Even though RNN-CTC models [6] eliminate the need for segmented inputs, they still need handcrafted features as an input, which becomes a bottleneck for such models. In

order to overcome this problem, multidimensional RNNs (MDRNNs) [7, 8] in combination with CTC as an end-to-end trainable model for detection of handwritten texts was proposed. The drawback of MDRNN models is their computational cost which can be alleviated by applying convolutional recurrent neural networks (CRNNs).

In CRNNs, the convolutional neural networks (CNNs) learn spatial and temporal information of characters within text images with less computational cost compared to MDRNNs. The state-of-the-art (SOTA) in many handwritten or printed text recognition tasks is the use of deep learning models consisting of CNNs as feature extractor and RNNs for sequence encoding by utilizing CTC as a loss function [9–12]. In literature, attention network mechanisms and transfer learning techniques were used to enhance the performance of CRNN based models [13–15]. Recently, encoder-decoder based sequence-to-sequence learning algorithms, which are widely used for language translation tasks, have also shown competitive results for HTR tasks [16–18].

A lot of HTR researches have been carried out for widely spoken languages, including English, Chinese, Japanese, Indian, and Arabic [19–22]. Despite the availability of large amount of handwritten text data for Amharic language, only a few attempts were made for recognition of its handwritten text. Assabie and Bigün [1] used a standard hand-engineered feature to represent character symbols and used HMM model to recognize handwritten words. The other related works considered isolated Amharic handwritten character recognition [23–26]. Recent studies in [27, 28] used LSTM-RNNs in combination with CTC algorithms for an end-to-end Amharic Optical Character Recognition(OCR). However, to the best of our knowledge, there is no study on offline Amharic handwritten text recognition using deep learning techniques. The absence of deep learning based approaches which has been successful in handwritten text recognition task of other languages motivated us to explore their applicability for handwritten Amharic word recognition. In addition, the absence of public handwritten Amharic text dataset motivated us to prepare the first publicly available dataset which will enable other researchers to compare and evaluate their techniques directly with others on similar data.

The contributions of this study are the following.

1. We designed custom CNN model and investigated different state-of-the-art CNN models, including DenseNet-121, ResNet-50 and VGG-19 with modification to fit our problem domain, for robust feature extraction from handwritten Amharic words images.
2. There is no public dataset for handwritten Amharic text recognition. In this study, we make available the

first public Amharic handwritten word image dataset called HARD-I. The dataset can be used for future studies to solve this problem and compare related studies.

3. We have conducted extensive experiments to evaluate the performance of four CRNN-CTC-based recognition models by analyzing different combinations of CNN and RNN network architectures, different input word image sizes and considering the effect of data augmentation. We have achieved state of the art recognition accuracy for handwritten Amharic words using our proposed method.

The rest of the paper is organized as follows: Sect. 2 presents a detailed description of the proposed method, including the dataset used in this study. In Sect. 2.3, model training and hyper-parameter optimization techniques are presented. In Sect. 2.4, we describe the performance evaluation metrics used in this study. The experimental results and discussions are given in 3. Finally, in Sect. 4, we present our concluding remarks.

2 The proposed methods and materials

In this section, we describe the proposed Amharic handwritten word recognition (AHWR) system. Figure 2 shows the general block diagram of the proposed end-to-end AHWR system used in this study. We will describe in detail the major components of the proposed models in the following subsections including the dataset preparation.

2.1 The proposed model architectures

The proposed model comprises three major components: CNN based feature extractors, RNN based sequence modeling, and CTC based output text transcription as shown in Fig. 2.

2.1.1 CNN-based feature extraction

In this study, we have done experiments to analyze the performance of different CNN architectures for extracting robust features for the Amharic handwritten word recognition task. We have studied four CNN-based feature extraction techniques in this study. These are custom model, VGG-19, ResNet-50, and DenseNet-121, which are CNN-based models. The application of these architectures requires careful consideration of their hyper-parameters for efficient AHWR tasks. The hyper-parameter choices are carried out using experiments on the training and validation datasets described in Sect. 2.2. After training the

Fig. 2 General overview of the AHWR system used in this study.

models, the best performing parameters of these architectures are selected as illustrated below.

We modified VGG-19, ResNet-50, and DenseNet-121 CNN model architectures by removing the top n -layers until we obtain $height(H) \times width(W) \times channel(C)$ feature map at the last convolution layer. As the CNN layer becomes more deeper, the down-sampled feature maps will have higher semantic information but the projected receptive field of the feature maps will cover larger area on the original image. Decreasing width of the feature map too much due to down-sampling will cause overlapping of features for two or more adjacent characters, which degrades performance of the whole model. In order to prevent such feature overlapping problem, we experimentally selected optimal values for the height and width of the feature map at the last convolution layer. We have done this by modifying strides of the max pooling layers at different layers of the proposed network architectures.

We achieved improved performance for our proposed models with feature map height (H) of 1 and width (W) of 32 generated at the last CNN layer. The channel (C) size at the last CNN layer varies from architecture to architecture. In order to keep the final feature map height and width at their optimal values, we have modified strides at stage 2, stage 3 and stage 4 convolutional blocks from (2, 2) to (2, 1) in ResNet-50 model, pool size and strides from (2, 2) to (2, 1) in all transition layers of DenseNet-121 architecture and pooling sizes from stride (2, 2) to (2, 1) in the last three max-pooling layers of VGG-19. We have obtained a final output feature map of $(1 \times 32 \times 2048)$, $(1 \times 32 \times 1920)$ and $(1 \times 32 \times 512)$ using ResNet-50, DenseNet-121 and VGG-19 models, respectively. We transform the final output feature map obtained from the CNN models into sequential features using column-wise concatenation. Then, we obtain a

sequential feature with sequence length of 32 and feature vector size of $(H \times W)$ at each time step, which is 2048, 1920 and 512 for ResNet-50, DenseNet-121 and VGG-19 respectively.

Our custom model consists of 12 convolutional layers with different number of filters ranging from 64 to 512 as depicted in Table 1. Max-pooling layers are added after the second, fourth, sixth, eighth and tenth convolution layers. In the last three max-pooling layers we set the pooling size to (2,1) in order to keep the width of the feature map at the last convolution layer at 32. We have used Batch Normalization [29] to resolve the internal covariate shift and rectified linear unit (ReLU) [30] at each convolution unit as a no-linear activation function. The output of the final feature map is reshaped into a feature vector of size 512 with sequence length of 32. Detail description of our custom CNN model is shown in Table 1.

2.1.2 RNN-based sequence modeling

RNNs are well known for learning temporal information in sequential data. In this work, we employ RNNs to encode sequential context information of input word images extracted by using our CNN models. We have used special type of RNNs known as long-short term memory (LSTM) [31] and gated-recurrent unit (GRU) [32] which have the capability to capture long range temporal information. We specifically used bidirectional LSTMs (Bi-LSTMs) and GRUs (Bi-GRUs), which have distinctive features to learn contextual information in the past and future using their memory-based network architecture. We have compared different combinations of these two RNN architectures by combining the two or using them separately, varying the number of layers

Table 1 Description of AHWR custom CNN model

Layer type	Description	Output size
Input	Binary word-image	$32 \times 128 \times 1$
Conv + ReLU + BN	Kernel 3×3	$32 \times 128 \times 64$
Conv + ReLU + BN + MaxPool	Kernel 3×3 , pool 2×2	$16 \times 64 \times 64$
Conv + ReLU + BN	Kernel 3×3	$16 \times 64 \times 128$
Conv + ReLU + BN + MaxPool	Kernel 3×3 , pool 2×2	$8 \times 32 \times 128$
Conv + ReLU + BN	Kernel 3×3	$8 \times 32 \times 256$
Conv + ReLU + BN + MaxPool	Kernel 3×3 , pool 2×1	$4 \times 32 \times 256$
Conv + ReLU + BN	Kernel 3×3	$4 \times 32 \times 256$
Conv + ReLU + BN + MaxPool	Kernel 3×3 , pool 2×1	$2 \times 32 \times 256$
Conv + ReLU + BN	Kernel 3×3	$2 \times 32 \times 512$
Conv + ReLU + BN + MaxPool	Kernel 3×3 , pool 2×1	$1 \times 32 \times 512$
Conv + ReLU + BN	Kernel 3×3	$1 \times 32 \times 512$
Conv + ReLU + BN	Kernel 2×2	$1 \times 32 \times 512$
Map-to-Sequence	Column-wise concatenation	32×512
Bi-GRU + dropout	512 hidden cells	32×1024
Bi-GRU + dropout	512 hidden cells	32×1024
Project (301 characters)	Project onto 301 classes	32×301
CTC	Decode or Loss calculation	≤ 32

Conv convolutional layer, MaxPool max pooling layer, BN batch normalization layer

and their hidden unit size. We have achieved optimal performance across all experiments when we use them separately rather than combining the two architectures. Two-layer architectures of both Bi-LSTM and Bi-GRU with a hidden unit size of 512 and dropout at each layer show better recognition performance. The AHWR model works slightly better using Bi-GRU since our dataset is not very large.

2.1.3 CTC based transcription

CTC based transcription is used in problems like ours where there is no one-to-one correspondence between the input and output sequences. The CTC algorithm takes as an input probability distributions of all the 301 symbols used in our study which are outputted from softmax activation layer at the end of the RNN sequence modeling unit and finds the maximum probable label sequence for a given input word image. The output probability distributions from the softmax layer are used by the CTC to calculate the conditional probability of a possible alignment path (π) given an input sequence x . The conditional probability $p(\pi|x)$ of a given alignment path is calculated by the product of all labels including blank character occurred at each time step along that path.

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

where π^t is the label occurred at time t , $y_{\pi_t}^t$ is probability of the label, and T is the input sequence length.

Paths should be mapped into correct label sequences since alignment paths may include a blank symbol which is not part of the desired label sequence and they may also hold repeated symbols during prediction. Thus a mapping function F is needed to remove repeated symbols and blanks providing correct label path for the set of aligned paths. For example, if we have an estimated path rr-ooo-ddd it will be mapped to rod but if we have path with alignment rr-o-oo-dd will be mapped to rood after removing the blanks and repeated characters. After the label is generated using the mapping function, conditional probability ($p(l|x)$) of a given mapped label l is calculated using sum of probabilities of all the paths mapped to that label using Eq. 2.

$$p(l|x) = \sum_{\pi \in F^{-1}(l)} p(\pi|x) \quad (2)$$

Given a training set consisting of input and label sequences $D = (x, l)$, the objective of the CTC loss function is to minimize the negative log probability in order to label correctly the entire training set data. The CTC loss function is expressed in-terms of the conditional probability of the labels as

$$CTC \text{ loss} = - \sum_{(x,l) \in D} \log p(l|x) \quad (3)$$

Then we can calculate the divergence of the loss function with respect to the output and weight parameters to train the handwritten recognition task in an end-to-end manner.

2.2 Dataset preparation

One of the main reasons for lack of researchs on HTR for Amharic language is unavailability of public datasets, which also becomes a challenge for researchers to directly compare their works with existing work. In this study, we have prepared Amharic handwritten word recognition dataset, which we called it Handwritten Amharic text Recognition Dataset (HARD-I), to solve these problems. During data acquisition, we prepared documents to be written by native speakers of the language. The documents used during data collection were the Aristotle monograph (translated by Hailegiorgis Mamo, wisdom from Pilate), Astronomy news letter (columnist: Zemene Yohannes, Zoskales), Ethiopian FDRE constitution, and Thus Spake Zarathustra (author: Friedrich Nietzsche; translated by Esubalew Amare). The above documents are used to collect diversified data and include many words from different domains (Table 2).

During data collection, 60 writers were asked to write textual content of the documents listed above with their free handwriting style. The individuals were from different age groups and educational backgrounds. After data acquisition, the handwritten document was scanned using a scanner with 1080 DPI and stored as a PNG file. We have used A* path-planning line segmentation [33], and scale-space word partitioning algorithms [34] during line and word segmentation pre-process step. The segmented word images were converted in to binary image with white pixels as background and black pixels as characters. Each word image is labeled manually based on the text it holds using UTF-8 character encoding scheme. The maximum word label length is 11 in our dataset. On the other hand, each word's label was encoded to numerical values based on their index in the Amharic character and symbol list as shown in Table 3.

The dataset contains 33,672 handwritten Amharic word images with their corresponding labels. From the total 33,672 word images, 12,064 are original handwritten images. The remaining 21,608 are augmented images generated by randomly applying functions such as rotation, shifting, shrinking, expanding, degrading, and applying a varying amount of Gaussian noise and blurring to the original handwritten image. Specifically, we have generated two augmented word images for each original handwritten word image. The augmentation is done only on the training and validation datasets, which is 90% of the total original handwritten dataset. The remaining 10% of

the total original handwritten dataset is used for testing the recognition models in all our experimental setups. Figure 3 shows sample pre-processing and augmentation techniques during data preparation process. The summary of the original handwritten dataset is shown in Table 2. The dataset is available for researchers through the link.¹

2.3 Model training and hyper-parameter optimization

In all the experiments, the proposed end-to-end AHWR models are trained using stochastic gradient decent (SGD) approach. We experimentally selected an initial learning rate of 0.0001 with Adam optimizer [35] and an epoch of 100 with mini-batch size of 8 to train the models. For all recognition models two different input word image sizes of 32(height) $\times$ 128 and 64(height) $\times$ 256 are selected experimentally to explore the effect on models' recognition performance. To improve performance of the recognition models and to avoid problem of over-fitting we also present our experimental analysis by training the recognition models using augmented data.

We have designed different models for AHWR task which we call them Model A, Model B, Model C and Model D. All these models have similar sequence modeling unit with two layers of Bi-GRU with 512 hidden units each and a dropout unit of 0.5 added at each GRU layers. They also have similar CTC loss and decoding function. These models differ from one another on the CNN feature extraction part. We have used our custom CNN, ResNet-50, DenseNet-121 and VGG-19 as a feature extractor in Model A, Model B, Model C, and Model D respectively.

We have conducted four different experiments for each of the recognition models. In the first two experiments, we have used the original handwritten dataset without augmentation using two different input word image sizes (i.e 32 $\times$ 128 and 64 $\times$ 256) while for the remaining two experiments we have used the augmented dataset using both input word image sizes. In all experiments, performance of the models is evaluated using our test dataset taken from the original handwritten dataset as described in Sect. 2.2. We used Keras framework with Tensorflow backend during model development. The training is carried out in Google cloud service called Google Colaboratory which provides a NVIDIA Tesla T4 GPU with 12 GB of RAM.

¹ <https://sites.google.com/view/hawdb-v1>.

Fig. 3 An example of data pre-processing and augmentation techniques used during handwritten Amharic word dataset preparation

Table 2 Detail description of HARD-I dataset

Data sources	Pages	Lines	Words
Aristotle monograph	7	178	1389
Astronomy newsletter	7	188	1476
Ethiopian constitution	50	1,074	7817
Thus Spake Zarathustra	6	163	1382
Total	70	1574	12,064

2.4 Performance evaluation metrics

We have used different metrics in order to evaluate performance of the different models used in our Amharic handwritten word recognition task. The most commonly used metrics for handwritten text recognition task are word error rate (WER) and character error rate (CER). The WER defines the number of words which are incorrectly predicted compared with the ground truth word labels. CER defines the minimal Levenshtein edit operations between predicted text and the ground truth transcription of the

word image. In addition, we have evaluated performance of the recognition models in terms of their number of training parameters and average recognition speed for a given word image.

3 Result and discussions

In this study, we have done extensive experiments to evaluate the performance of the proposed handwritten Amharic word recognition models. The proposed models are evaluated using the HARD-I dataset described in Sect. 2.2. We used 80% of the data for training, 10% for validation and the remaining 10% for testing purposes.

Our experimental results are shown in Table 4. In our experimental design using the original handwritten dataset without augmentation, Model C which is based on DenseNet-121 as feature extractor outperforms all other recognition models on both word image sizes. It achieves a CER of 2.61% and WER of 9.08% using 32×128 input word image size and a CER of 1.99% and WER of 6.75% using 64

Table 3 Amharic characters and other symbols used in this study.

0	ሀ	15	ሐ	30	ሠ	45	ረ	60	ሸ	75	ብ	90	ቱ	105	ታ	120	ኛ	135	ኪ	150	ወ	165	ዘ	180	ኸ	195	ዲ	210	ኅ	225	ጠ	240	ዳ	255	ፋ	270	ኀ	285	1
1	ሁ	16	ሑ	31	ሡ	46	ሰ	61	ሿ	76	ቦ	91	ቲ	106	ቴ	121	ኼ	136	ካ	151	ዐ	166	ዛ	181	ኹ	196	ዳ	211	ኆ	226	ጡ	241	ደ	256	ፊ	271	ሀ	286	2
2	ሂ	17	ሒ	32	ሢ	47	ሱ	62	ቸ	77	ባ	92	ታ	107	ት	122	ኽ	137	ኼ	152	ዐ	167	ዘ	182	ኸ	197	ዲ	212	ኅ	227	ጠ	242	ዳ	257	ፋ	272	ሀ	287	3
3	ሃ	18	ሓ	33	ሣ	48	ሲ	63	ቹ	78	ባ	93	ቲ	108	ታ	123	ኽ	138	ኪ	153	ዐ	168	ዛ	183	ኹ	198	ዳ	213	ኆ	228	ጡ	243	ደ	258	ፊ	273	ሀ	288	4
4	ሄ	19	ሔ	34	ሤ	49	ሳ	64	ቺ	79	ባ	94	ቲ	109	ታ	124	ኼ	139	ኪ	154	ዐ	169	ዛ	184	ኹ	199	ዳ	214	ኆ	229	ጡ	244	ደ	259	ፋ	274	ሀ	289	5
5	ህ	20	ሐ	35	ሥ	50	ሲ	65	ቻ	80	ባ	95	ቲ	110	ታ	125	ኼ	140	ኪ	155	ዐ	170	ዘ	185	ኸ	200	ዳ	215	ኆ	230	ጡ	245	ደ	260	ፋ	275	ሀ	290	6
6	ሆ	21	ሑ	36	ሦ	51	ሲ	66	ቼ	81	ባ	96	ቲ	111	ታ	126	ኼ	141	ኪ	156	ዐ	171	ዘ	186	ኹ	201	ዳ	216	ኆ	231	ጡ	246	ደ	261	ፋ	276	ሀ	291	7
7	ለ	22	መ	37	ሢ	52	ሱ	67	ቻ	82	ባ	97	ቲ	112	ታ	127	ኼ	142	ኪ	157	ዐ	172	ዘ	187	ኹ	202	ዳ	217	ኆ	232	ጡ	247	ደ	262	ፋ	277	ሀ	292	8
8	ሉ	23	መ	38	ሢ	53	ሲ	68	ቻ	83	ባ	98	ቲ	113	ታ	128	ኼ	143	ኪ	158	ዐ	173	ዘ	188	ኹ	203	ዳ	218	ኆ	233	ጡ	248	ደ	263	ፋ	278	ሀ	293	9
9	ሊ	24	ሚ	39	ሢ	54	ሸ	69	ቻ	84	ባ	99	ቲ	114	ታ	129	ኼ	144	ኪ	159	ዐ	174	ዘ	189	ኹ	204	ዳ	219	ኆ	234	ጡ	249	ደ	264	ፋ	279	ሀ	294	10
10	ላ	25	ሚ	40	ሢ	55	ሸ	70	ባ	85	ባ	100	ታ	115	ታ	130	ኼ	145	ኪ	160	ዐ	175	ዘ	190	ኹ	205	ዳ	220	ጡ	235	ጡ	250	ደ	265	ፋ	280	ሀ	295	11
11	ላ	26	ሚ	41	ሢ	56	ሸ	71	ባ	86	ታ	101	ታ	116	ታ	131	ኼ	146	ኪ	161	ዐ	176	ዘ	191	ኹ	206	ዳ	221	ጡ	236	ጡ	251	ደ	266	ፋ	281	ሀ	296	...
12	ል	27	ሞ	42	ሢ	57	ሸ	72	ባ	87	ታ	102	ታ	117	ታ	132	ኼ	147	ኪ	162	ዐ	177	ዘ	192	ኹ	207	ዳ	222	ጡ	237	ጡ	252	ደ	267	ፋ	282	ሀ	297	*
13	ሎ	28	ሞ	43	ሢ	58	ሸ	73	ባ	88	ታ	103	ታ	118	ታ	133	ኼ	148	ኪ	163	ዐ	178	ዘ	193	ኹ	208	ዳ	223	ጡ	238	ጡ	253	ደ	268	ፋ	283	ሀ	298	#
14	ሊ	29	ሚ	44	ሢ	59	ሸ	74	ባ	89	ታ	104	ታ	119	ታ	134	ኼ	149	ኪ	164	ዐ	179	ዘ	194	ኹ	209	ዳ	224	ጡ	239	ጡ	254	ደ	269	ፋ	284	0	299	?

Table 4 Recognition performance comparison of AHWR models on our test dataset, with and without augmentation using two different input word image sizes.

Model	Augmentation	32×128 input image size		64×256 input image size	
		CER (%)	WER (%)	CER (%)	WER (%)
Model A		3.14	11.17	3.66	13.00
	✓	1.98	6.58	2.65	9.25
Model B		3.39	11.42	2.75	8.66
	✓	2.58	8.5	1.54	5.67
Model C		2.61	9.08	1.99	6.75
	✓	2.19	7.25	1.15	5.24
Model D		10.96	27.58	3.51	12.58
	✓	5.46	18.25	5.43	17.67

× 256 input word image size. Model A shows better recognition performance compared with Model B and Model D using an input word image size of 32×128 . Model D shows the worst recognition accuracy compared with the other models on both input word image sizes. This can be attributed to deep-stacked convolution and max-pooling layers without any shortcut connection or feature concatenation like the other two deep-stacked models, ResNet-50 and DenseNet-121.

Table 4 also illustrates that training recognition models using augmented data have shown significant improvement in recognition accuracy of the models, as compared to using original handwritten dataset without augmentation. This is due to the increased number of training images, enabling the models to learn more robust features by preventing overfitting and intra-class variation between vocabulary symbols. Model A outperforms the other models with a CER of 1.98% and WER of 6.58% using an input word image size of 32×128 . On the other hand, using input word image size of 64×256 Model C, which uses DenseNet-121 as feature extractor, outperforms all the other models, and it achieved the best recognition performance with a CER of 1.15% and WER of 5.24% for the AHWR task.

From our experimental results, we have seen that DenseNet-121 and ResNet-50 architectures show better recognition performance than VGG-19 due to their network architecture design which avoids the problem of vanishing gradients. The selection of an optimal size of the input word image has a significant effect on the performance of the handwritten text recognition task. When we use small input word image sizes, shallow network architectures extract more relevant feature representation of words, while increasing the size of input word images requires deeper network architecture for better feature representation. Data augmentation has led to improved

Table 5 Average recognition speed and number of trainable parameters of different recognition models used in this study.

Model	# of Parameters		Recognition speed in s/image	
	32×128	64×256	32×128	64×256
Model A	17,462,893	19,035,757	$1.22e^{-3}$	$2.40e^{-3}$
Model B	36,432,045	36,432,045	$1.20e^{-3}$	$2.10e^{-3}$
Model C	24,628,461	24,628,461	$3.20e^{-3}$	$8.40e^{-3}$
Model D	28,208,365	29,781,229	$1.42e^{-3}$	$2.62e^{-3}$

performance of the models by increasing the training data variation.

The performance of the models in terms of their recognition speed and number of training parameters is also evaluated as shown in Table 5. The recognition speed indicates the average time required to recognize a given word image during inference. Even though Model B using Resnet-50 as feature extractor has a large number of training parameters, it has fast recognition speed compared to the other models. This is due to its thin and deep network architecture design, where convolution filters grow slowly. Model C using DenseNet-121 as feature extractor and with fewer training parameters than Model B has shown slower recognition speed due to its very deep network architecture with more convolution layers, which contributes to computationally expensive operations. Despite its shallow network, Model D with VGG-19 as feature extractor is also computationally expensive than Model B due to a large number of convolution operations on the first two layers. Our custom model shows comparable recognition speed to Model B with the smallest number of trainable parameters than all the other models.

We have also shown examples of recognition results on sample word images taken from our test dataset for visual analysis in Fig. 4. The green text indicates the correct predictions while the red text indicates wrongly predicted characters. For deleted characters, we use underscore “_” to represent their location. From Fig. 4, we can see that for randomly selected four handwritten Amharic word images, our recognition models predict wrongly by deleting, substituting or inserting some of the characters in the transcribed text. In Amharic language, many characters have almost similar shapes due to the formation of derived characters from base character with little modification. This similarity in shape becomes one of the challenges for our recognition models to recognize characters correctly. Apart from that, most of the errors occur due to disconnected character parts during hand-writing and cursive nature of handwritten text, which causes a character shape difference compared with the ground

holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Assabie Y, Bigün J (2011) Offline handwritten Amharic word recognition. *Pattern Recognit Lett* 32:1089–1099
- Giménez A, Juan A (2009) Embedded Bernoulli mixture HMMs for handwritten word recognition. In: 2009 10th international conference on document analysis and recognition, pp 896–900
- Marti U-V, Bunke H (2001) Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *IJPRAI* 15(65–90):02. <https://doi.org/10.1142/S0218001401000848>
- Senior AW, Fallside F (1992) Off-line handwriting recognition by recurrent error propagation networks. In: David H, Roger B (eds) *BMVC92*. Springer, London, pp 453–461. ISBN 978-1-4471-3201-1
- Graves A, Fernández S, Gomez F, Schmidhuber J (2006) Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on machine learning, ICML '06. New York, NY, USA, pp 369–376. Association for Computing Machinery. ISBN 1595933832. <https://doi.org/10.1145/1143844.1143891>
- Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2009) A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 31(5):855–868
- Voigtlaender P, Doetsch P, Ney H (2016) Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In: 2016 15th international conference on frontiers in handwriting recognition (ICFHR), pp 228–233
- Graves A, Schmidhuber J (2009) Offline handwriting recognition with multidimensional recurrent neural networks. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) *Advances in neural information processing systems 21*. Curran Associates, Inc., pp 545–552. <http://papers.nips.cc/paper/3449-offline-handwriting-recognition-with-multidimensional-recurrent-neural-networks.pdf>
- Puigcerver J (2017) Are multidimensional recurrent layers really necessary for handwritten text recognition? In: 2017 14th IAPR international conference on document analysis and recognition (ICDAR), vol 01, pp 67–72
- Safarzadeh VM, Jafarzadeh P (2020) Offline Persian handwriting recognition with CNN and RNN-CTC. In: 2020 25th international computer conference, computer society of Iran (CSICC), pp 1–10
- Tran H-P, Smith A, Dimla E (2019) Offline handwritten text recognition using convolutional recurrent neural network, pp 51–56, 11. <https://doi.org/10.1109/ACOMP.2019.00015>
- Zhang X, Yan K (2019) An algorithm of bidirectional RNN for offline handwritten Chinese text recognition. In: Huang D-S, Huang Z-K, Hussain A (eds) *Intelligent computing methodologies*. Springer, Cham, pp 423–431
- Chowdhury A, Vig L (2018) An efficient end-to-end neural model for handwritten text recognition. In: CoRR. <http://arxiv.org/abs/1807.07965>
- Granet A, Morin E, Mouchère H, Quiniou S, Viard-Gaudin C (2018) Transfer learning for handwriting recognition on historical documents. In: 7th international conference on pattern recognition applications and methods (ICPRAM), Madeira, Portugal. <https://hal.archives-ouvertes.fr/hal-01681126>
- Chamchong R, Gao W, McDonnell MD (2019) Thai handwritten recognition on text block-based from Thai archive manuscripts. In: 2019 International conference on document analysis and recognition (ICDAR), pp 1346–1351
- Kang L, Toledo JL, Riba P, Villegas M, Fornés A, Rusiñol M (2019) Convololve, attend and spell: an attention-based sequence-to-sequence model for handwritten word recognition. In: Thomas B, Andrés B, Mario F (eds) *Pattern recognition*. Springer, Cham, pp 459–472
- Wei H, Liu C, Zhang H, Bao F, Gao G (2019) End-to-end model for offline handwritten Mongolian word recognition. In: Jie T, Min-Yen K, Dongyan Z, Sujian L, Hongying Z (eds) *Natural language processing and Chinese computing*. Springer, Cham, pp 220–230. ISBN 978-3-030-32236-6
- Geetha R, Thilagam T, Padmavathy T (2021) Effective offline handwritten text recognition model based on a sequence-to-sequence approach with CNN-RNN networks. *Neural Comput Appl*. <https://doi.org/10.1007/s00521-020-05556-5>
- Parvez M, Mahmoud S (2013) Offline Arabic handwritten text recognition: a survey. *ACM Comput Surv (CSUR)* 45:02. <https://doi.org/10.1145/2431211.2431222>
- Srihari S, Yang X, Ball GR (2007) Offline Chinese handwriting recognition: an assessment of current technology. *Front Comput Sci China* 1:137–155
- Sujatha P, Bhaskari L (2019) A survey on offline handwritten text recognition of popular Indian scripts. *Int J Comput Sci Eng* 7:138–149
- Trier ØD, Jain Anil K, Taxt T (1996) Feature extraction methods for character recognition—a survey. *Pattern Recognit* 29(4):641–662. [https://doi.org/10.1016/0031-3203\(95\)00118-2](https://doi.org/10.1016/0031-3203(95)00118-2)
- Gondere MS, Schmidt-Thieme L, Boltana AS, Jomaa HS (2019) Handwritten Amharic character recognition using a convolutional neural network
- Abdurahman F (2019) Handwritten Amharic character recognition system using convolutional neural networks. *Eng Sci* 14:71–87
- Meshesha M, Jawahar C (2007) Optical character recognition of Amharic documents. *Afr J Inf Commun Technol*. <https://doi.org/10.5130/ajict.v3i2.543>
- Cowell J, Hussain F (2003) Amharic character recognition using a fast signature based algorithm. In: Proceedings on seventh international conference on information visualization, 2003. IV 2003, pp 384–389. <https://doi.org/10.1109/IV.2003.1218014>
- Belay B, Tewodros H, Meshesha M, Marcus L, Gebeyehu B, Stricker D (2020) Amharic OCR: an end-to-end learning. *Appl Sci* 10:1117
- Belay B, Habtegebrial T, Liwicki M, Belay G, Stricker D (2019) Amharic text image recognition: database, algorithm, and analysis. In: 2019 International conference on document analysis and recognition (ICDAR), pp 1268–1273
- Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: CoRR. <http://arxiv.org/abs/1502.03167>
- Andrew L (2013) Maas. Rectifier nonlinearities improve neural network acoustic models
- Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2017) LSTM: a search space odyssey. *IEEE Trans Neural Netw Learn Syst* 28(10):2222–2232. <https://doi.org/10.1109/TNNLS.2016.2582924>
- Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation
- Surinta O, Holtkamp M, Karabaa F, Van Oosten JP, Schomaker L, Wiering M (2014) A path planning for line segmentation of handwritten documents. In: 2014 14th international conference

- on frontiers in handwriting recognition, pp 175–180. <https://doi.org/10.1109/ICFHR.2014.37>
34. Manmatha R, Rothfeder JL (2005) A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Trans Pattern Anal Mach Intell* 27(8):1212–1225. <https://doi.org/10.1109/TPAMI.2005.150>
 35. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.