Amharic Character Recognition Using Deep Convolutional Neural Network

Achamie Aynalem(✉)

Bahir Dar Institute of Technology, Bahir Dar, Ethiopia
aynacham@gmail.com

Abstract. Amharic is the working language in the Federal Democratic Republic of Ethiopia. The Amharic alphabet has a large number of symbols and there is a close resemblance among shapes of the different symbols available in the language which challenged the task of machine-based optical character recognition systems in the language. The absence of a standardized labeled dataset for the Amharic language created additional barriers for different researchers. Our aim in this paper is to design a deep convolutional neural network based architecture that could extract features and classify Amharic characters with significant confidence of accuracy that could be utilized for real-world applications. A total of 90,000 characters are prepared for training the proposed architecture and an additional of 25,000 characters are reserved for testing purpose. Due to the occurrence of a large number of symbols and a close resemblance in the shapes of the different characters available in the language, a relatively complex convolutional Neural Network is utilized to capture those features and categorize them into the correct characters. Dropout layers are utilized to avoid overfitting. The character recognition system proposed in this paper achieved an accuracy of 99.27% on the testing dataset which is a significant improvement for the Amharic language. The implementation was done using Tensorflow on Keras neural network layers and Opencv in python to pre-process image data which enables us to make the system readily available for software developers as an API.

Keywords: Amharic recognition · Character recognition · Deep convolutional neural networks · Deep learning application

1 Introduction

Amharic which is one of widely spoken Semitic language is the working language in the Federal Democratic republic of Ethiopia. Alphabets of the Amharic language which are also called Ethiopic alphabet is an indigenous writing alphabet used for writing different languages in Ethiopia including Amharic, Geez, Tigrinya, Agaw and several other languages in Ethiopia, Eritrea and northern parts of Sudan. The Amharic alphabets are believed to have been derived from the ancient Geez language which is today confined to only for church services in the Ethiopian and Eritrean Orthodox and Catholic churches.

B. H. Woldegiorgis et al. (Eds.): ICAST 2022, LNICST 455, pp. 177–195, 2023.
https://doi.org/10.1007/978-3-031-28725-1_11

Different historical, political, socio-cultural and academic documents are found written in different languages using the Ethiopic alphabet.

Many of the ancient scientific, historical and sacred books of Christianity and Judaism are also found on different parts of the world written in the Geez language using the Ethiopic alphabet [1].

In this decade, researchers have made a major breakthrough in creating deep learning based models for character recognition mainly on languages such as English, Chinese and some other Latino based languages. However, there is no much research on languages which use the Ethiopic alphabet like Amharic and Tigrinya [2].

Amharic alphabet has large number of symbols and there is a close resemblance among shapes of the different symbols available in the language which made machine based character recognition systems of the language challenging. The absence of a standardised labelled dataset for Amharic language created additional barriers for different researchers. In this research paper, we have proposed a model for Amharic character recognition system using Deep Convolutional Neural Network. The model has multiple convolutional layers that could capture the variations in the features of closely resembling characters in the language.

1.1 Features in the Amharic Alphabet

The Amharic has 34 base characters. Unlike English where different sounds of a consonant are written with different combinations of the five vowels, in Amharic language, the different sounds of a base character are represented with different symbols. There are seven basic vowels in Amharic which are shown in the following figure (Fig. 1).

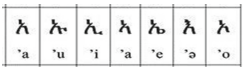

Fig. 1. The seven vowels in Amharic

The language has an additional 8th vowel which is not used quite often, but available in conservative writers of the language and in other languages like Geez and Tigrinya.

Each of the 34 base characters produce a total of 34 * 8 = 272 symbols which represent the different sounds generated by combination of those base characters with the eight vowels. A partial view of the Amharic character set is shown in Fig. 2 below.

Taking the Ethiopic base character (hä)as an example, we can generate the eight symbols derived from this base character. The 8 symbols generated with combination of this base character with the 8 vowels are (hä), (hu), (hee), (ha), (hae), (heh), (ho)and (hua). In addition to these alphabetic symbols, there are more than 10 punctuation marks and 20 numeric symbols resulting more than 300 symbols in the language.

In addition to the large number of distinct symbols in the language, most of the Ethiopic alphabets have a higher degree of morphological structure similarities making the recognition process much difficult to recognize the characters as expected. Sometimes, even for our own eyes, it might be difficult to differentiate accurately among

Fig. 2. A partial view of the Amharic alphabet

slightly deformed characters in the language. We can take an example of " ለ" and " ሰ", " ደ" and "ጸ", " ገ" and " ነ", or " ከ" and " አ" which looks alike. The different symbols for vowel combinations of a base character is usually derived by adding small strokes at the right of the base or introducing a small structural variation on the base character.

For example consider the base character, ለ(le)whose eight derivations are ለ(le), ሉ(lu), ሊ(li), ላ(a), ሌ(lĕ), ል(lĭ), ሎ(lo)and ሏ(lua).

As we can see, there is a close resemblance in the morphological structure among most of the eight derivative symbols of the base character. Image processing algorithms should be carefully designed during recognition of Amharic characters in order not to remove or erode these little extensions or strokes available in the alphabet.

1.2 Offline Character Recognition System

The process of character recognition involves the conversion of images containing handwritten or printed texts into machine-readable text format.

Character recognition complexity depends on the distinct shapes, strokes and the number of characters available in the language.

In general, there are two types of text recognition approaches: off-line and on-line systems. Off-line text recognition system involves converting already existing scanned image or image captured through digital camera into text; whereas on-line text recognition system involves converting the different strokes and lines emanating from a real time stream of data captured through different transducers such as electromagnetic or pressure sensitive touch pens of tablet into sequences of characters.

In offline character recognition system the document is first generated, digitized, stored in computer hard disk and then it is processed latter. It is not real a time process.

Off-line handwritten character recognition refers to the process of recognizing characters in a document that have already been scanned or captured through a digital camera which might be from a sheet of paper or a label plate and are then stored in digital format image.

The character recognition process involves extracting different features from an image containing characters; and approximating some groups of extracted features to the nearest resembling character based on a previously "learned knowledge" of feature-to-character map.

1.3 The Deep Convolutional Neural Network

In the recent years, deep neural networks brought key breakthrough in different application areas like computer vision, image recognition, natural language processing and in speech recognition. In the past decade, deep networks have enabled machines to recognize images, speech and even play games at accuracy even impossible for humans. Likewise, a number of Artificial Neural Network Based methods have been used for character recognition for different languages.

Morphological/Rank/Linear Neural Network (MRL-NN) [3], where the different combinations of inputs in every node is formed by hybrid linear and nonlinear (of the morphological/rank type) operations, was studied for handwritten digit recognition. A hybrid Multilayer Perceptron Support Vector Machine (MLP-SVM) model was used for recognition of English numerals [4] and Chinese character [5] recognition. Support Vector Machine (SVM) with Radial Basis Function (RBF) network was also used for character recognition of English language [6].

In the last decade, Convolutional Neural Networks (CNN) is found efficient for handwritten character recognition due to its capability of capturing spatial information [7].

CNN exploits local spatial correlation by introducing local connectivity constraints between neurons of adjacent layers, which made CNN very well-suited for classification problems. Convolutional Neural Networks also provides some degree of translational invariance which made them further suitable for extracting and classifying patterns from an image. A CNN based model has shown a significant improvement on the accuracy of English character recognition. A CNN based model was tested on UNIPEN [6] English character dataset and found recognition rates of 93.7% and 90.2% for lowercase and uppercase characters, respectively [8].

A Deep Convolutional Neural Network (DCNN) contains multiple neural network layers. There are particularly three different types of layers used in convolutional neural network model; these are the convolutional layers, pooling layers and fully connected layers as shown in Fig. 3 below.

The Convolutional Layer: This layer is used to extract local features from matrix that come from an input image using different filters. This mimics receptive neurons in biological nervous system of living organisms. The convolutional layer will enable us to extract different features from an input image by utilizing several filters parallely.

Pooling Layers, (aka down sampling). Performs dimensionality reduction, which involves reducing the number of parameters from the input by applying an aggregation function to the values generated at the convolutional layer. Two types of pooling filters or matrices are available; max pooling and average pooling filters.

Fully Connected Layer: The last stage in a convolutional neural network is commonly made of one or more fully connected layers. The fully connected layer performs classification of features that are extracted from the input image at using the previous layers of the network. The convolutional and pooling layers usually utilise the ReLu activation functions for extraction of features from the input image, whereas, fully connected layers usually implement a softmax activation function which always produce either 0 or 1 for classification of features extracted at the previous stages.

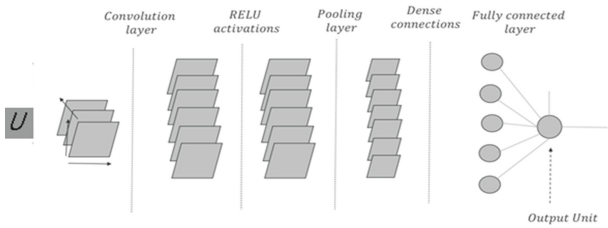

Fig. 3. Components of a convolutional Neural Network

In this paper, we have developed a DCNN based deep learning model for recognition of Amharic text.

Deep learning algorithms depends heavily on data. Different machine learning algorithms use different dataset and the dataset is highly specific to the particular model. Dataset is one of the most crucial aspects that makes algorithm training possible and it highly affect the performance of a machine learning algorithm. To achieve high level of accuracy, large amount of dataset is needed for training and henceforth powerful computing power is needed to train deep neural networks. Data preparation is one of the most difficult steps in most machine learning project.

There is no sufficient public accessible dataset for Amharic character recognition. In the absence of such dataset, we have prepared a dataset of Amharic characters collected from different news and magazines published in Amharic language.

During the dataset preparation, first we have collected the images that contain the desired characters using a scanner. After the image is accessed, different image processing algorithms were used to improve quality of the image; then line and character segmentation algorithms were used to separate segments of images that contain characters. These segments of pixels that represent characters are finally stored as a matrix of those fragments within sub folders with labels that represent the classes of each fragment.

The DCNN is used to classify those segments to the correct characters and produce a binary encoded output for each class of characters by comparing them to an already trained state of the network.

Finally those binary coded outputs of the model are post-processed to produce human understandable outputs such as decoding binary coded outputs to human readable symbols, representing character spacing, word spacing and line breaking. However, the post classification process is not implemented in this research paper in order not to scatter quite broad idea in a single paper.

The basic phases of data preparation and classification stages are shown in the following Fig. 4.

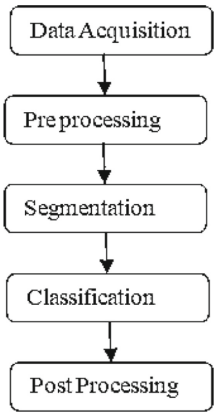

Fig. 4. Block diagram of the character recognition system

2 Data Acquisition

In this research we have collected more than 120 pages of Amharic document from different fiction books, Addis Zemen magazine, Ethiopian church documents from www.eotmk website and from Ethiopian Code of Civil rights. Those documents were also synthetically reformatted with different font families available in Amharic in order to generate images of characters with different shapes.

3 Image Pre-processing

Most of the time the image captured through a camera or scanner may not have the required quality for extracting features that represent characters. Therefore, first the acquired image should pass through a series of image pre-processing algorithms in order to enhance the quality of images required for preparing a dataset.

3.1 Grey Scale Conversion

Image captured through camera or scanner has always three channels which represent the Red, Green and Blue channels of the image. Since colour does not have significant information content except adding some aesthetic value to the document, the RGB scale input image should be converted to a gray scale image which will subsequently reduce the processing time and complexity of the neural network used for the recognition process.

3.2 Noise Removal

Image acquired through scanner or digital cameras is usually associated with some noise due to factors like malfunctioned devices, misalignment of paper, poor light intensity during image capturing or scanning,

Different filters such as median filter and Gaussian filter with different kernels are tested for removing noise which could be approximately modelled with salt and paper noise. However, due to a common feature of thin horizontal extensions available in Amharic alphabet, those filters have the tendency to average out those extensions to the background pixel or degrade them (Fig. 5).

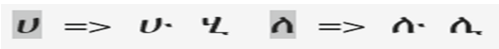

Fig. 5. Thin horizontal extension in Amharic alphabet

An adaptive Gaussian filter with an asymmetric kernel resulted in relatively better performance for removing such salt and paper noise with reduced degrading effect on the lateral extensions. In order to minimize this erosion of small lateral extensions, we have used asymmetric kernel of (5, 3) for the Gaussian filter. Relatively small value is deliberately used in the x-axis which enabled us to preserve those small horizontal strokes commonly available in the language.

3.3 Binary Image Conversion

Once the images are converted to grayscale and the irrelevant noises are removed from the images, the image is converted to binary scale image where the image is converted into binary images so that the image is composed of pixels having a colour value of either black (pixel value of 0) or white (pixel value of 255). An adaptive threshold filter is used to classify pixels into white or dark value adaptively depending on the local variance using a kernel size of 5. The [5] kernel is used to select local matrix from the region along the image surface and a threshold filter is applied on this local matrix to classify pixels to either 0 (text pixel) or 1 (background pixel) depending on a local threshold value. The kernel or window is then slided horizontally and vertically with the step size until the whole pixel available within the image are classified to 0 or 1. This could effectively classify pixels into dark and white values from an image with variable spacial background intensity since the threshold value is calculated locally depending on the local pixel variance. The input image as shown in Fig. 6/A has background with variable intensity along the vertical axis. This variable background is effectively removed using the adaptive threshold as shown in Fig. 6/B below.

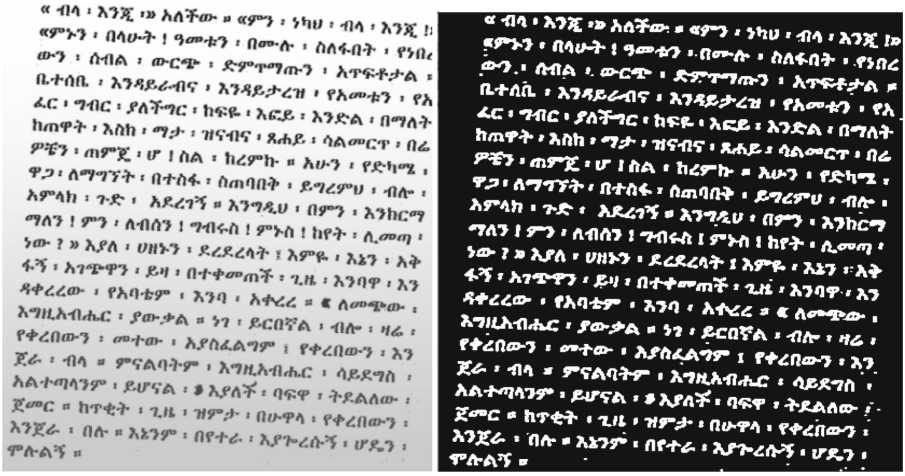

Fig. 6. A) Input Image with variable background. B) Binary scale conversion using adaptive threshold followed by background inversion.

During this adaptive threshold binarization process, there are noise introduced into the image due to misclassification of some background pixels as text pixels which result in small black pixels randomly distributed across the image. Therefore, the resulting binary image is further passed through Gaussian noise in order to remove such salt and paper noise introduced during the binarization process. A global, inverted threshold algorithm is then used to correct smoothening or blurring effect introduced by the Gaussian filter and to invert the colour scale such that the background is represented with black (pixel values of 0) and the foreground or characters with white (pixel values of 255).

3.4 Skew Correction

Sometimes due to misalignment of paper during the scanning process, or while capturing through camera, lines of the text are rotated randomly with small angle. This creates sever problem in subsequent stages during segmentation of lines of characters from the image. A skew detection and correction algorithm is utilized. The algorithm first finds all the pixel coordinates that are part of the foreground image. Then the rotation angle of the rectangle is calculated based on the collected pixel coordinates. Then by using the calculated rotation angle, we will find the rotation matrix with respect to the centre coordinates of the image which will be used to rotate the input binary image as shown in Fig. 7.

Fig. 7. Skew corrected image

3.5 Segmentation

Segmentation is the process of breaking the whole image into subparts to process them further. Segmentation is a critical step in the handwritten recognition process and it highly affects the accuracy of the recognition process. There are three levels of segmentation in character recognition.

Line Segmentation

This involves cropping each line of text out of the image and store them in a sorted manner. Thus, using those lines, words and characters can be extracted from each line with the original order. Horizontal histogram projection is used to segment the individual lines of text from the image.

During the horizontal projection

- Rows that represented a text pixel in a line have high number of foreground pixels, which corresponds to higher peak values in the histogram.
- Rows that represent the gaps in-between the lines have high number of background pixels, which correspond to lower peaks in the histogram.

Therefore, rows that correspond to lower peaks in the histogram can be selected as the segmenting lines to separate the lines.

Figure 8 shows vertically expanded view of the histogram of the first 9 lines. As shown in the figure, the picks of the histogram represent lines that represent text since text are represented with white pixels (255) whereas, pixels that represent gaps between lines will have minimum point in the histogram since there will not be text along those lines. Partitioning the image regions horizontally using a threshold pixel value could effectively segment the image into lines as shown in Fig. 9.

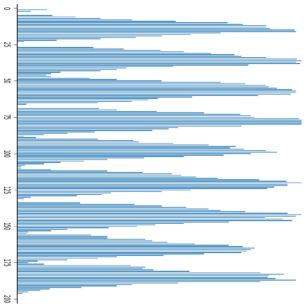

Fig. 8. Histogram of lines in horizontal projection

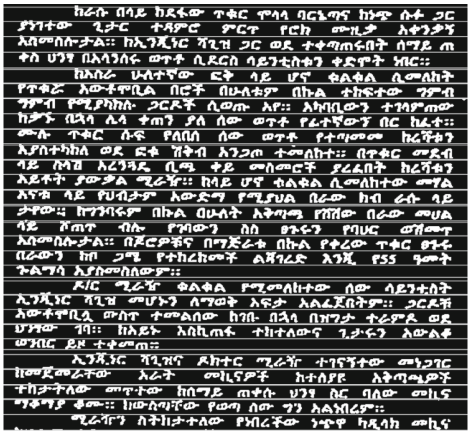

Fig. 9. Segmented lines

Word and Character Segmentation

Once the lines are segmented successfully word and character segmentation is carried out to crop out segments of each character from an image which contain a line of characters. Vertical histogram projection technique resulted in poor result during segmenting characters from within a line due to too narrow space in between characters within a word or due to overlapping pixel of neighbouring characters of a word. The problem is much worse in texts that contain oblique font styles. Consider Amharic text fragments ቡታ፣ ሙጋwhere the first character has a forward stroke, whereas the second character has backward structural extension which make the gap between the neighbouring characters zero. The problem becomes worse when the text is in italic font style resulting an overlapped regions between characters. This made isolating characters using vertical projection technique inefficient. For the character segmentation, we have used contour detection segmentation method which involve the following steps;

1. Accept the list of lines which are returned from the line segmentation stage.
2. Find all contours/characters on the line and return a list of coordinates for each contour.

3. Crop the characters from the threshold text line images using the returned coordinates.
4. Iterate through each line.

Fig. 10. Segmented characters

This could segment characters of a line into separate chunks as shown in Fig. 10.

4 Dataset Preparation

Due to existence of relatively large number of symbols and resemblance in the morphological shapes among the different characters available in the language, we have designed relatively complex convolutional Neural Network with significantly larger number of trainable parameters which would be able to map features available in the language. Such neural network models with large number of parameters require large and diversified dataset in order to be able to capture those large number of diverse features and categorize them into the correct character; and avoid over fitting of the model which is a cause of degraded performance of a model on a real-world data.

However, there is limited public accessible datasets for character recognition in Amharic language. There is one dataset prepared by Birhanu Hailu Belay which contains the first 231 synthetically generated text-line images with Power Geez and Visual Geez fonts with each image 32 * 32 size. The dataset does not contain all characters available in the language; and numeric symbols and punctuation marks are not included, either.

Therefore, we have prepared an additional dataset that contain Amharic characters, numerals and punctuation marks. We have collected 105 pages of Amharic text collected from Amharic fiction books, Addis Zemen Gazetta and Ethiopian Code of civil Rights and from Ethiopian Orthodox church documents. Those documents are reformatted using different Amharic font families and segmented into 32 * 32 separate character segments to generate a dataset which contain most of the characters, numeric symbols and punctuation marks in Amharic language. This dataset then merged with the dataset which is found from the aforementioned source and then shuffled and augmented to a representative dataset.

5 Data Augmentation

Data segmentation is a set of techniques to artificially increase the amount and quality of data by generating new slightly modified data points from the existing data. Data augmentation techniques enable machine learning models to be more robust by creating variations that the model may came across in the real world. We have increased size and "quality" of the dataset using 5° rotation, adding some Gaussian noise and rescaling

with a ±5% factor. The resulting dataset is shuffled to produce a randomly distributed dataset.

This dataset is finally separated into groups of their respective symbols, with images of different characters placed in different folders where folders named in an extended alphabetical order. Labels for each class of the dataset will be subsequently generated from folder labels using Tensorflow.

A total of 90,000 characters are prepared for training the proposed architecture and an additional 25,000 characters are reserved for testing purpose.

6 DCNN Architecture for Recognition

Naturally, the number of trainable **parameters available in a neural network model is proportional** to the **complexity** of the task the model has to perform; requiring a **proportional amount of examples or labelled dataset**, to get good performance. In this research paper, the proposed model for recognition of Amharic characters is Deep Convolutional Network. A CNN primarily composed of two components: a feature extractor module which is composed of convolutional and pooling layers with a ReLu activation function followed by a trainable DNN classifier.

The convolutional neural network contains the following layers:

Input layer which accepts batches of image matrices of 32 * 32 size.
Convolution layer will take images from an input layer or from preceding layers and convolve its input with a specified number of filters to create feature maps. The number of feature maps extracted is equal to the specified number of filters used at a convolutional layer. Convolutional layers are generally used with the ReLu activation function.

As stated previously, the Amharic character set contains relatively large number of symbols with each symbol containing highly resembling morphological features. In order to be able to map such features, we have selected three stages of feature extraction with each stage extracting different depths of features. Each stage is consists of two convolutional layers separated with batch normalization layers followed by a maxpooling layer and dropout layer.
Pooling layer will down sample the 2D activation maps along the height and width dimensions. Pooling operators aggregate the values of a sub-matrix generated by a convolutional filter into one single output value that represent the "meaning" associated with that special region.
Dropout layer will regularize weights in the fully connected layers of a convolutional neural network and feed forward networks to avoid overfitting. Too much cooperation between neurons makes the neurons dependent on each other and they fail to learn distinct features which usually result in doing classification well on a training dataset but produce incorrect prediction if tested on somewhat different dataset, which is an overfitting problem.

In order to solve this overfitting problem in Deep Neural Networks, a specified proportion of neural network units is randomly dropped at training time for each training sample in a mini batch. This makes the remaining neurons learn important features all by themselves without relying on cooperation from other neurons. In this model, we

have used four dropout layers each in between the three stages and one in between the dense connected layer and the output layer.

We have used padding of 2 for both dimensions (width and height) at each convolutional layer; and maxpooling layers with a pool size of (2,2) at the end of each convolutional stage.

Fully connected layers are dense feed forward neural layers that receive different sets of weights from the preceding layers. Each neuron in this layer will be connected either to all the neurons in the previous layer or to all the neurons in the next layer. There are two fully connected layers; one hidden layer and the output layer. For classification, the output neurons receive inputs from the final hidden fully connected layers and generate the appropriate class for the particular input pattern.

We have considered 300 Amharic characters, punctuation marks and numeric digits; therefore the output layer should have 300 neurons which produce one-hot encoded output for each character in the Amharic language (Fig. 11).

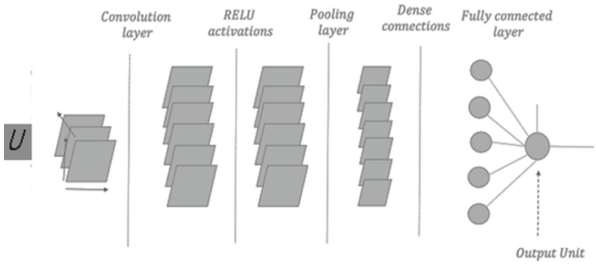

Fig. 11. A general structure of the convolutional Neural Network Model

7 Training CNN Model

The process of training a neural network is simply the process of updating the weights of the neural networks. Unlike, the Multi-Layer Perceptions (MLP), in CNN neurons share the same weights among them and also, they are followed with pooling layers. In which the sharing of weights among the neurons and pooling layers will help to decrease the overall weights of the neural network and computational power. Commonly an activation layer particularly sigmoid and ReLU activation function is placed between the convolutional and pooling layers. Once the image passes through these processes, the features from the image will be extracted and then unzipped into a 1D vector which will then be used by the successive dense layers of the network for classifying the input imag. Consider I is a two-dimensional image vector and K is filtering window which has a size of w*h, then the convolution process can be given as:

$$(1+K)_{ij} = \sum_{m=0}^{w-1} \sum_{n=0}^{h-1} km, n * Ii + m, j + n + b \tag{1}$$

The weights in the neural network architecture are calculated using this convolution operation which is why this architecture is named Convolutional Network.

Backpropagation:
It is one of the most commonly used training algorithms in neural network. The process of training a neural network requires a greater computational power so, we need to make our design and training process as efficient as possible. The process of training CNNs can be also called convolution operations which utilize the mathematical correlation operation. Training a neural model involves updating the weight and biase parameters available in the network. In the case of backpropagation, the weights are not the only things to be updated also the deltas should be updated.

The updates to the weights can be computed using the expression;

$$\frac{\partial E}{\partial w^l_{m',n'}}, \tag{2}$$

The gradient components of each weight can be calculated using a chain role which is expressed using the following mathematical expression.

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{H-k_2} \delta^l_{i,j} o^{l-1}_{i+m',j+n'} \tag{3}$$

$$\frac{\partial E}{\partial w^l_{m',n'}} = rot_{180^\circ}\{\delta^l_{i,j}\} * o^{l-1}_{m',n'} \tag{4}$$

The summation represents all the sum of the gradients from $\delta^l_{i,j}$ coming from the output layer l in which the double summation is for the shared weights of the filter kernel. The chaining role used in the above equation is used for the optimization of the backpropagation training process. As can be understood from this discussion, in the backpropagation training method, we use a chain role to optimize the training process of the CNN architecture. Stochastic Gradient Descent (SGD) algorithm utilize the Backpropagation in order to calculate the gradient or derivative of the loss function.

Then, after the inputs to each neuron is calculated using the forward propagation, the output of the network is used in backward fashion to update the weights based on the chosen loss function.

The total error of prediction of the different classes is calculated using the mean squared error equation or using cross entropy from the network output y for a true output t as shown in the following equations.

$$E = \frac{1}{2} \sum_p (t - y)^2 \tag{5}$$

$$E = -\sum_p y \log(t) \tag{6}$$

Training the DCNN was handled interactively using Tensorflow.

Table 1. Different parameters in the proposed model

NO_CLASS = 300

Layer (type)	Output Shape	No. of Parameters
Stage I Conv.		
conv2d_0 (Conv2D)	(None, 28, 28, 20)	200
normalizer_0 (Batch Normalization)	(None, 28, 28, 20)	80
conv2d_1 (Conv2D)	(None, 28, 28, 20)	3620
normalizer_1 (Batch Normalization)	(None, 28, 28, 20)	80
max_pooling_0 (MaxPooling2D)	(None, 14, 14, 20)	0
dropout (Dropout)	(None, 14, 14, 20)	0
Stage II Conv.		
conv2d_2 (Conv2D)	(None, 14, 14, 64)	11584
normalizer_2 Batch Normalization)	(None, 14, 14, 64)	256
conv2d_3 (Conv2D)	(None, 14, 14, 64)	36928
normalizer_3(Batch Normalization)	(None, 14, 14, 64)	256
max_pooling_1 (MaxPooling 2D)	(None, 7, 7, 64)	0
dropout_1 (Dropout)	(None, 7, 7, 64)	0
Stage III Conv.		
conv2d_4 (Conv2D)	(None, 7, 7, 128)	73856
normalizer_4 (Batch Normaliz'n)	(None, 7, 7, 128)	512
conv2d_5 (Conv2D)	(None, 7, 7 128)	147584
normalizer_5 (Bath Normaliz'n)	(None, 7, 7, 128)	512
max_pooling_2 (MaxPooling 2D)	(None, 3, 3, 128)	0
dropout_2 (Dropout)	(None, 3, 3, 128)	0
Dense Layer		
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 500)	576500
dropout_3 (Dropout)	(None, 500)	0
output (Dense)	(None, 300)	150300
Total parameters: 1,002, 268		

Output of the network parameters generated during the training process is shown in the following Table 1.

The input to the model is batches of images with their respective labels which contained dataset of the Amharic characters and one-hot encoded labels of each character. Each of the characters are reshaped to a size of 32 × 32 pixels. This relatively large pixel size is chosen in order not to lose small features differences between the different characters available in the language. The images are fed in batches as four-dimensional tensors where the first dimension is specific to the image index in the batch, second and third

dimensions are specific to the height and width of the image. The model contains three successive feature extraction stages. Each of those stages contains convolutional layers which could be able to extract different levels of features successively; and maximum pooling layer.

8 Experimental Result and Discussion

The experiments were conducted using Tensorflow with keraz neural layers in Python. The input of the network is a dataset of 32 * 32 sized images. Images for each symbol are stored in separate folders with each folder labelled in ascending order of the Amharic characters in the Unicode encoding order. Then, during the data reading stage, those images are read and one-hot encoded labels for each image are generated from folder names. Therefore, image fragments which are found within the same folder represent instances of the same character and are all associated with the same output label. This process of fetching data and associating them to output labels is handled in a separate python routine that generates training and testing datasets from input images and their labels is handled in a separate python routine.

The resulting dataset is passed to the DCNN model and trained using the 'Adam' optimizer algorithm, which took about one hour to train the model for 20 epochs on a local machine. The model took significantly longer time for the training process due to the large number of training parameters available in the model and the slow processor speed of the local machine. The output of the training process on the Pycharm terminal is displayed in the following table.

Table 2. Training the proposed model for 20 epochs

```
Epoch 1/20
271s 567ms/step - loss: 0.3892 - accuracy: 0.8875 - val_loss: 5.7527 - val_accuracy: 0.1126

Epoch 16/20
- 233s 534ms/step - loss: 0.0261 - accuracy: 0.9920 - val_loss: 0.0465 - val_accuracy: 0.9872
Epoch 17/20
- 231s 530ms/step - loss: 0.0263 - accuracy: 0.9921 - val_loss: 0.1024 - val_accuracy: 0.9760
Epoch 18/20
- 233s 534ms/step - loss: 0.0285 - accuracy: 0.9915 - val_loss: 0.0364 - val_accuracy: 0.9911
Epoch 19/20
- 231s 530ms/step - loss: 0.0253 - accuracy: 0.9923 - val_loss: 0.0309 - val_accuracy: 0.9925
Epoch 20/20
- 245s 570ms/step - loss: 0.0253 - accuracy: 0.9932 - val_loss: 0.0286 - val_accuracy: 0.9925

Do you want to save state? y/n: y
- 15s 111ms/step - loss: 0.0297 - accuracy: 0.9927
================================================================================
Test loss:  0.0292 - Test accuracy:  0.9927
================================================================================
```

The model was run tuned with hyper parameters shown in Table 2 above. The choice of the hyper parameters is empirical with a focus on the learning behavior of the model.

As shown in the following figures, the model rapidly converges around 8 epochs and gained only a little improvement in the accuracy after the 8th epoch. Therefore, around 25 min are required to train the model with a reasonable accuracy of prediction, which is achievable at the 8th epoch. The loss curve shows a drop from 5.75 to 0.0286 on the validation dataset and to 0.0292 on the test dataset; whereas the accuracy jumped to 99.32% on the training dataset and to 99.27% on the test dataset, as shown in Table 2 above.

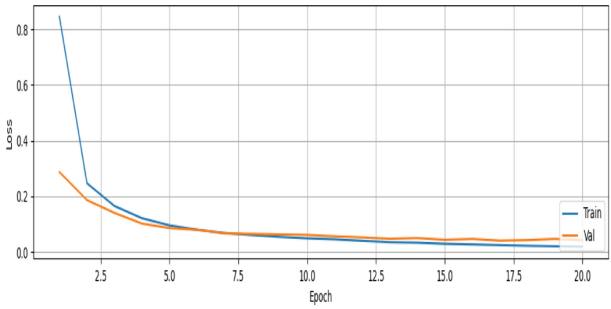

Fig. 12. Loss of the DNN model for 20 epochs

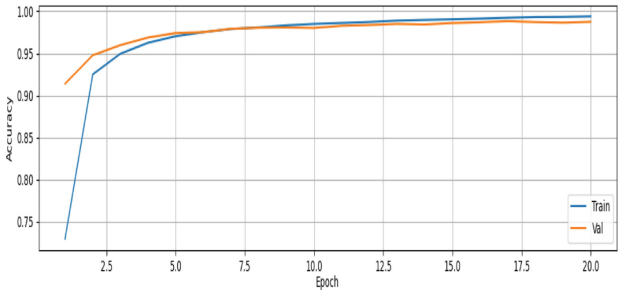

Fig. 13. Accuracy of the DNN model for 20 epochs

As we can see from the graphs at Fig. 12 and 13, it could achieve a prediction accuracy of 99.27% on the test dataset. This model misclassified around 7 characters out of a text containing 1000 characters, which is very good accuracy of prediction in comparison to previous models which have been proposed for Amharic recognition [10].

9 Conclusion and Future Work

The main objective of this research is to design a CNN based character recognition architecture for the Amharic language. The research work included image pre-processing techniques that could be utilized for the implementation of the system for real world

applications. We have prepared a dataset merged with an existing database to generate a representative dataset in further research on the recognition of the Amharic language. The dataset contains 90,000 training images and 25,000 testing images consisting of symbols of most of the Amharic characters, punctuation marks and numeric digits.

As we can see from Table 2 shown above, we could achieve an accuracy of 99.27% on the test dataset. This model misclassified less than 7 characters from a text containing 1000 characters. In contrast to previous works by different individuals where most of the works on recognition of Amharic or Geez language which have been done using MATLAB, in this Amharic text recognition system, image pre-processing and recognition algorithms are all implemented using Tensorflow 2.x with tf.keras network models using the python language. This would enable us to create an API for android, desktop-based or web application software that needs to integrate recognition of Ethiopic languages.

We have compiled a representative dataset and will try to make it available by putting it on public accessible repositories for further research on Ethiopic languages recognition.

References

1. Yemisrach, B.: Ethiopian historical antiques (1999). Retrieved 10 May 2022
2. Chirag, I.P., Ripal, P., Palak, P.: Handwritten character recognition neural network. Int. J. Sci. Eng. Res. **2** (2011)
3. Int. J. Eng. Comput. Sci. **4**(5), 11729–11732 (2015)
4. Pessoa, L.F.C., Maragos, P.: Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition. Pattern Recogn. **33**(6), 945–960 (2000)
5. Bellili, A., Gilloux, M., Gallinari, P.: An MLP-SVM combination architecture for offline handwritten digit recognition. IJDAR **5**, 244–252 (2003)
6. Dong, J., Krzyżak, A., Suen, C.Y.: An improved handwritten Chinese character recognition system using support vector machine. Pattern Recogn. Lett. **26**(12), 1849–1856 (2005)
7. Theodoridis, S., Koutroumbas, K.: Pattern recognition and neural networks. In: Paliouras, G., Karkaletsis, V., Spyropoulos, C.D. (eds.) ACAI 1999. LNCS (LNAI), vol. 2049, pp. 169–195. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44673-7_8
8. Guyon, L., Schomaker, R., Plamondon, R., Liberman, M., Janet, S.: Unipen project of on-line data exchange and recognizer benchmarks. In: Proeedings. of 12th International. Conference on Pattern Recognition (ICPR), vol. 2, pp. 29–33. IEEE (1994)
9. Yuan, A., Bai, G., Jiao, L., Liu, Y.: Offline handwritten English character recognition based on convolutional neural network. In: 10th IAPR International Workshop on Document Analysis Systems (DAS), pp. 125–129 (2012). https://doi.org/10.1109/DAS.2012.61
10. Siranesh, G., Menore, T.: Ancient Ethiopic manuscript recognition using deep learning artifcial neural network (Unpublished master's thesis). Addis Ababa University (2016)
11. Ahangar, R.G., Ahangar, M.F.: Handwritten farsi character recognition using artificial neural network. Int. J. Comput. Sci. Inform. Security **4** (2009)
12. Christos, S.: Dimitrios, S.: Neural Networks. https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.htmlpril (2018)
13. Ahmad, I., Fink, G.: Class-based contextual modeling for handwritten Arabic text recognition. In: 2016 15th international conference on frontiers in handwriting recognition (ICFHR) (2016)

14. Shafi, M.: Optical character recognition of printed Persian/Arabic documents. https://scholar.uwindsor.ca/etd/5179 (2014). Retrieved 16 Dec 2018
15. Kavallieratou, E., Sgarbas, K., Fakotakis, N., Kokkinakis, G.: Handwritten word recognition based on structural characteristics and lexical support. In: 7th International Conference on Document Analysis and Recognition (2003)
16. Nasien, D., Haron, H., Yuhaniz, S.S.: Support vector machine (SVM) for english handwritten character recognition. In: 2010 Second International Conference on Computer Engineering and Applications, Bali, Indonesia, pp. 249–252 (2010)