Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2022, Article ID 8515810, 12 pages
https://doi.org/10.1155/2022/8515810

Research Article

Handwritten Geez Digit Recognition Using Deep Learning

Mukerem Ali Nur, Mesfin Abebe, and Rajesh Sharma Rajendran

Adama Science and Technology University, Adama, Ethiopia

Correspondence should be addressed to Rajesh Sharma Rajendran; sharmaphd10@gmail.com

Received 29 June 2022; Revised 26 September 2022; Accepted 5 October 2022; Published 8 November 2022

Academic Editor: Abidhan Bardhan

Amharic language is the second most spoken language in the Semitic family after Arabic. In Ethiopia and neighboring countries more than 100 million people speak the Amharic language. There are many historical documents that are written using the Geez script. Digitizing historical handwritten documents and recognizing handwritten characters is essential to preserving valuable documents. Handwritten digit recognition is one of the tasks of digitizing handwritten documents from different sources. Currently, handwritten Geez digit recognition researches are very few, and there is no available organized dataset for the public researchers. Convolutional neural network (CNN) is preferable for pattern recognition like in handwritten document recognition by extracting a feature from different styles of writing. In this work, the proposed model is to recognize Geez digits using CNN. Deep neural networks, which have recently shown exceptional performance in numerous pattern recognition and machine learning applications, are used to recognize handwritten Geez digits, but this has not been attempted for Ethiopic scripts. Our dataset, which contains 51,952 images of handwritten Geez digits collected from 524 individuals, is used to train and evaluate the CNN model. The application of the CNN improves the performance of several machine-learning classification methods significantly. Our proposed CNN model has an accuracy of 96.21% and a loss of 0.2013. In comparison to earlier research works on Geez handwritten digit recognition, the study was able to attain higher recognition accuracy using the developed CNN model.

1. Introduction

Amharic language is the only African language with its own alphabet and writing system while most of the other African languages use Latin and Arabic alphabets for their own writing system [1]. The Federal Democratic Republic of Ethiopia and other regional states use the Amharic language as their official working language. It is the mother language for over 50 million people and the second language for over 100 million people in Ethiopia [1]. Arabic is the only Semitic language spoken more than Amharic in the world. Amharic is also spoken by some people in neighboring countries like Eritrea, Djibouti, and Somalia. There are many historical documents written in Geez scripts found in Ethiopia. There are around 80 different languages spoken in Ethiopia, with up to 200 dialects. The Geez alphabet is used as the writing system in some languages. Amharic, Geez, and Tigrinya are the most spoken languages in Ethiopia that use the Geez alphabet [1].

Geez script consists of 265 characters including 27 labialized characters (characters representing 2 sounds), 20 symbols for numerals, and 8 punctuation marks [2]. Our research focused on only the Geez digits. Geez numerals have been used in Ethiopian calendars, Geez Bibles, and historical documents. Geez numbers consist of twenty different symbols to represent the numerical values. Unlike Latin numbers, 0 is not represented by any symbol. Twenty numbers are represented by independent symbols such as 1–9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, and 10000 as shown in Figure 1. Other numbers are represented by the combination of those twenty symbols. Each digit symbol has a dash (horizontal line) above and below the digit character.

Handwritten character and digit recognition works are done in different languages to improve the efficiency of the recognition when they digitize historical and handwritten documents [4]. Digit recognition is a well-known problem that has been used to document indexing using dates such as document date, birth date, marriage date, and death date [5].

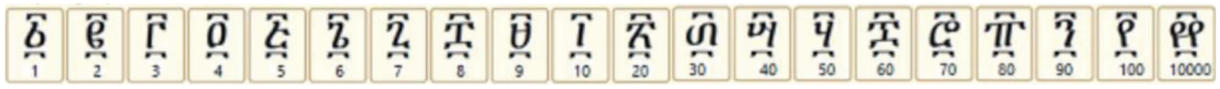

Figure 1: Geez number representations [3].

Digit recognition and detection have been utilized in a variety of applications, including automated the reading of the number of bank cheques, postal numbers and codes, tax forms, and document indexing based on dates [6]. There are two types of architectures for handwritten digit string recognition. The two strategies for recognizing the digit string are detection-free and segmentation-based recognition [7]. In segmentation based on the system, we first detect the numerical string that may contain multiple digits. Splitting digits should be done before a recognition to isolate each digit [8, 9]. However, detection-free recognition approach recognizes each digit without any splitting and detection preprocesses [10].

Random Forest, SVM, KNN, and other machine learning techniques have been developed to recognize handwritten digits. Deep learning methods like CNN have the highest accuracy when compared to the most commonly used machine learning algorithms for handwritten digit recognition [11, 12]. Pattern recognition and large-scale image classification are both done with CNN. Handwriting character recognition is a research field in computer vision, artificial intelligence, and pattern recognition [1]. It might be claimed that a computer application that conducts handwriting recognition has the capacity to acquire and recognize characters in photographs, paper documents, and other sources, and convert them to electronic or machine-encoded form. Deep learning is a popular field of machine learning that uses hierarchical structures to learn high-level abstractions from data. According to references [13, 14], the availability of technology CPUs, GPUs, and hard drives, among other things, machine learning algorithms, and large data, such as MNIST handwritten digit data sets and ImageNet data, are all factors in deep learning's success. Handwritten digit recognition, facial recognition, computer vision, audio and visual signal analysis, voice recognition, disaster recognition, and automated language processing are all areas where deep learning is applied [15].

Nowadays, deep learning is becoming a popular technique to learn to recognize patterns and deep patterns and extract. It has a deep learning level to generate patterns from a given dataset. It is an amazing algorithm with diverse libraries to extract patterns and recognize from images and classify them. Among the deep learning algorithms, the CNN is efficient and has good image classification, image recognition, pattern recognition, feature extraction, and so on.

2. Related Works

Kusetogullari et al. [5] introduced a deep learning architecture known as DIGITNET to detect and recognize English handwritten digits that are found in historical documents in Sweden. The authors also created a large-scale handwritten digit dataset for the public known as DIDA. The data were collected from the Swedish handwritten historical documents written by different priests in the nineteenth century. The dataset consists of 100,000 handwritten digit images. The DIGITNET consists of two different architectures to detect a digit and recognize the digit. The first architecture is DIGITNET-dect which detects the digit strings from handwritten documents and the second architecture is DIGITNET-rec which recognizes the handwritten digit. The authors used a deep learning approach to train both models and used regression-based deep CNN methods to detect the digit. YOLOv3 was designed by the authors to detect and classify a digit from an image. In the recognition phase, three different CNN architectures were proposed by the authors. Convolutional, batch normalization, max-pooling, fully-connected layers, and SoftMax layers are all included in each proposed model. But still, it has a limitation of some of the image data having high resolution, so it increases the computational cost in the training of the model and some digits are not labeled due to their bad appearance. Low digit detection accuracy because of negative sampling is also a limitation of the research work.

Chen et al. [16] compared five machine learning classification models to recognize handwritten digits offline. The authors compared the performance of the KNN, neural network, random forest, decision tree, and bagging with gradient boost. 70,000 digit images are used to develop the classifier models. The KNN and neural network show better accuracy than other classifiers and KNN achieves 10 times faster speed than the neural network model. The preprocessing stage is the crucial part of the recognition system in handwritten recognition. The authors used some preprocessing techniques to enhance the data. They used normalization to give equal weight to each attribute. Then, they used a median filter for the noise reduction step. Image sharpening and image attribute reduction are the other steps in the preprocessing phase, but still, it has some limitations from those, the bewilder tool is not effective to preprocess handwritten image data and they did not find a threshold value for the binarization preprocess technique; then, they ignore binarization technique. The image is blurred after median filter and sharpening in preprocessing techniques.

Beyene [3] proposed a multilayered feed-forward propagation ANN for offline handwritten and machine-printed Amharic (Geez) number recognition. The author collected only 560 datasets for the model. He used 460 for the training and 100 for the test data. The author collected the data manually because there is no public data for Geez handwritten digits. The overall classification accuracy is 89.88%, which is poor because he used a very small amount of the data to develop his model [3]. Many researchhas been experimented in the specific area of handwritten digits of ancient Semitic language (Geez). Some other researchers have done for all Geez character recognition but the author [3] did his research specifically on Geez digits. But still, it has some limitations from those, a small number of data are used to train the algorithm, the work does not give any information about the preprocessing technique, and the accuracy of the proposed model is low to recognize the digit. Hossain and Ali [17] proposed a handwritten digit recognition using a CNN on MNIST handwritten datasets. The authors used MatConvoNet to increase the speed of the operation of building the proposed model. MatConvoNet is a MATLAB function that supports an efficient computation on CPU and GPU allowing the training of complex models on large datasets such as Image Net ILSVRC. However, it has some limitations such as the research does not give any information about the preprocessing technique and the number of hidden convolution layers is small in the proposed model.

Demilew and Sekeroglu [1] proposed an ancient Geez script recognition model by using deep learning. The authors developed a deep CNN model to recognize Ethiopian ancient Geez characters found in historical documents. They proposed an architecture that only recognizes Geez characters and not words or full sentences. The dataset is a total of 22,913 images collected from libraries, private books, and the Ethiopian Orthodox Tewahedo Church. They also developed a recognition system to recognize twenty-six base characters only. In Geez scripts, there are around 265 characters and 34 base characters, but they classified each character to its base character class, not to its specific character. There are 7 characters found in each base class including the base class. One of the challenges in recognizing handwritten Geez script is the similarity between the characters which are found in the same base class. The authors classified all of the seven characters found in the same class into one base class and ignored the difficult task in their model, but still, it has the problem of low image quality, the number of instances is not balanced for each character. Also, the research work does not mention the methods that are used for character detection. The proposed model classified all of the seven characters found in the same class into one base class; this is the other limitation.

Gondere et al. [2] designed a handwritten Geez character recognition system using a CNN. The authors used multitask learning to enhance the model from the relationships of the characters. They ran the experiment by some hyper-parameters of a CNN. The parameters are 100 batches in size, 0.3 keeping probability for dropout, 0.0001 learning rate, and 0.01 L2 regularization. They organized a dataset from different previous research works. But still, it has some problems in the research work. The first one is they used a unique handwritten dataset that affected the performance of the models and the work does not mention the preprocessing technique. Ali et al. [18] proposed a model to recognize a handwritten digit. The authors used a CNN algorithm to develop the model. They used deeplearning4j with a CNN for the recognition system. The CNN is composed of two main tasks. The first task is to extract a feature from each layer. Each layer takes input from the output of the previous layer and forwards the current output to the next layer. The second task of the CNN architecture is feature classification. This unit generates or classifies the predicted output. The authors used the MNIST dataset for their work. 60,000 handwritten digit images were used for training and testing the model. But it has some limitations from those, the proposed model used a large kernel size in the convolution layer, and because of that, it consumes a longer training time. Also, the work does not give any detailed information about the preprocessing technique.

Most of the researchers did digit recognition on English numbers. They achieved high performance using different methods to recognize handwritten digits. For English handwritten digits, there are many resources and datasets ready to be used by the research community. It encourages the researchers to focus on that area. However, for Geez handwritten digits, there are no organized data in public for researchers to work on recognition of handwritten digits. Some researchers did Geez character recognition for machine-printed and handwritten characters but they did not focus on digits, especially for handwritten. The author of [3] is the first researcher to work on recognizing handwritten Geez digits, but the dataset he used was a very small and low performance made.

3. Data Collection Method

For this study, handwritten data were collected from a variety of people with various writing styles. Instead of manual feature extraction, which is difficult for humans to do, deep learning models are utilized, which are life-simplifying and efficient techniques to extract with high accuracy, and performance. A data-gathering paper was created for this purpose. The data gathering paper is prepared in a way to make the pre-processing easier. The paper is A4 size which consists of the symbol of all 20 Geez numbers, in 2 rows and 10 columns in a box, and other same-sized empty boxes prepared and repeated 5 times as shown in Figure 2. This means an individual has to handwrite 100 instances or digits. The data were collected from 524 different individuals and each person gave 100 instances of digits. According to calculations, since the collected data are from 524 different individuals, 52,400 instances are obtained. People from many demographic groups participated in the data collection. The data were gathered from elementary pupils, high school students, high school staff members, university students, and university academic staff (lecturers). The majority of information was acquired from university students, which totaled roughly to 250 at Adama Science and Technology University.

The data collection in the university was successfully conducted with the help of Computer science and Engineering Club ASTU (CSEC-ASTU) members. The club had 100 members at the time of data collection; thus, the data were gathered from them and through their connections on the campus. As mentioned earlier, data were obtained from 250 university students, 150 of whom are male and 100 of whom are female. After collecting the data, it must be converted from paper to digital format before it can be processed. The documents were scanned using a TECNO mobile with a 50 Mega Pixel camera and a software app called cam scanner for this process. The advantage of using a

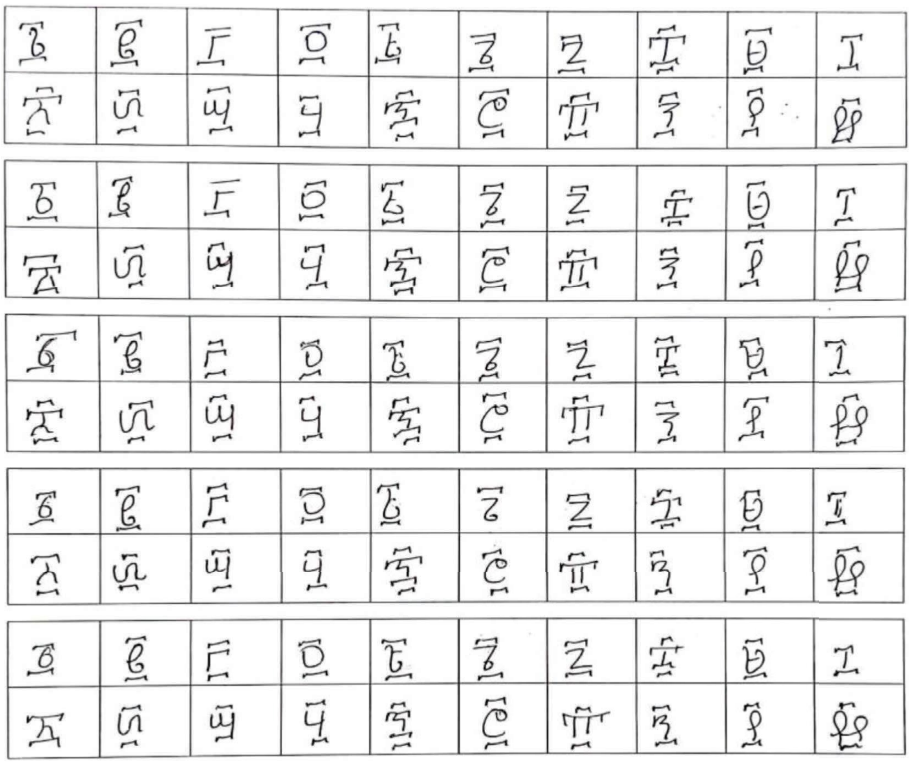

Figure 2: Sample handwritten Geez digit data.

cam scanner is that it detects the paper and provides only the digital format (in image format) of the paper part after removing the background, reducing noise.

Python's OpenCV library was used for data extraction during the pre-processing technique. This program's input is one partition, and its output is the extracted data. Once prepared for one partition, the same would go for others.

4. Data Preprocessing

The second phase of the proposed model is the preprocessing phase that occurs after the digital image has been made. The digitized image is first checked for skewing before being preprocessed to reduce noise. Preprocessing is necessary for creating data that are simple to recognize using handwritten digit recognition systems, and the goal is to reduce background noise, enhance the image's region of interest, and produce a clear distinction between foreground and background. The study use the Python OpenCV library for the preprocessing technique.

4.1. Resize Image. Because the data are available in a range of sizes, it must be resized to fit the network's input size. All images are resized to 32×32 pixels in this work. This scaling is important for reducing computational complexity and for concentrating on the region of interest by cropping it.

4.2. RGB to Grayscale Conversion. The simplest color model is grayscale, which specifies colors using only one component: lightness. A value ranging from 0 (black) to 255 (white) is used to define the amount of brightness (white). All the original images in the dataset are in RGB color format. Converting the RGB to grayscale, reduce the color channel, and it reduces the computational complexity compared with RGB color images. In our proposed model the input images are grayscale so, the original images should be converted to the grayscale color format.

4.3. Color Inversion. The dominant color of the original image is white, which has a value of 255. For grayscale image, dataset models changing the dominant color to black is preferable to reduce the complexity of the mathematical operations. Because black color has 0 values, a convolution operation with the dominant part with a 0 value is reducing the computational complexity of the model. Figure 3 shows the preprocessing techniques used in our dataset. As shown in Figure 3(d), the dominant part of the image is the background of the image. In the color inversion technique, the background is converted from white to black color as shown in Figure 4.

5. Proposed Model

The convolutional neural network (CNN) is the proposed model to address the Geez handwritten digit recognition. To recognize the digits, a CNN-based digit classifier is used. Six different CNN-based handwritten digit classifiers consist of a number of layers such as a convolutional layer, max-pooling layer, dropout layer, flatten layer, fully-connected layers, and SoftMax layer to achieve high recognition

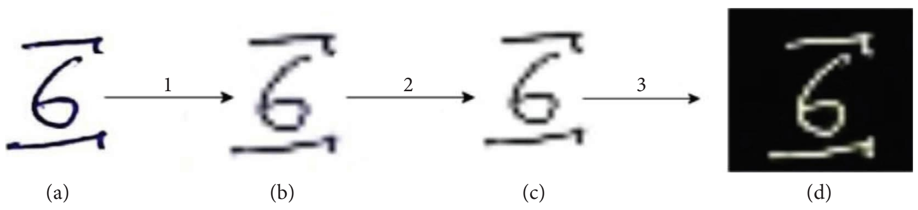

Figure 3: Geez handwritten digit image after some preprocessing techniques. (a) Original image (b) Resize image. (c) Grayscale image. (d) Black background image.

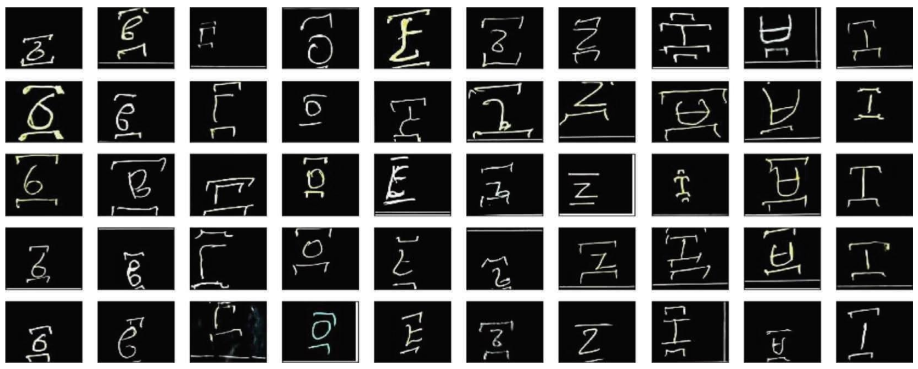

Figure 4: Sample images of the Geez handwritten digit dataset.

accuracy. Furthermore, the training was performed by applying the backpropagation approach of stochastic gradient descent.

Finally, based on the evaluation metric, choose the best model for recognizing digit strings. Each classifier is constructed with a different number of convolutional layers, kernel sizes, and filters. The parameters applied in all six classifiers are summarized in Table 1. Model 6, for example, shown in Figure 5 has 8 convolutional layers, 4 max-pooling layers, 3 dropout layers, 2 fully connected layers, and 20 output layers. The kernel size, stride, and number of filters in the first convolutional layer are 3×3, 1, and 32 (3×3@1@32), respectively. The second and third convolution layers are similar to the first. After three convolution layers, the max-pooling layer (2×2@2@32) is applied. The convolutional layer (3×3@1@64) is used in the fifth layer, and it consists of 64 filters with a kernel size of 3 3 and a stride of 1. The following two layers are convolution layers, with the same hyperparameter as the fifth layer. The max-pooling layer (2×2@2@64) is applied in the eighth layer. After the max-pooling layer, dropout is applied. The convolutional layer (3×3@1@64) is applied next, which consists of 64 filters with a kernel size of 3 3 and a stride of 1. The max-pooling layer (2×2@2@64) is the next hidden layer.

After the max-pooling layer, the dropout is applied. The convolutional layer (3×3@1@128) is applied next, which consists of 128 filters with a kernel size of 3×3 and a stride of 1. The max-pooling layer along with dropout layer is used before the fully connected layer. Fully connected layers are used, which consist of 128 nodes. In the convolutional and fully connected layers, ReLU is used as an activation function. SoftMax is used as a last layer to compute the probabilities of output classes in the last layer. The class with the highest probability produces the desired result. The epoch size is 30 and the total number of training instances in a single batch is 32. The other five classifiers have varying numbers of convolutional and fully connected layers, as well as different layer organizations. The first fully connected layer contains 128 neurons and the second contains 20 neurons for all cases.

6. Result and Discussion

The CNN is used to observe and see the differences of the accuracies among different results from the handwritten Geez digit models. Training and validation accuracy were measured for 30 different epochs by changing out hidden layers for various combinations of convolution layers and using batch size 32 in all cases. Figures 6, 7, 8, 9, 10, and 11 illustrate the accuracy of the CNN, and Figures 12, 13, 14, 15, 16, and 17 show the loss of the CNN with various convolution and hidden layer combinations. Table 1 shows the maximum and minimum training and validation accuracies of the CNN determined after experiments for six different cases with different hidden layers, and Table 2 shows the maximum and minimum training and validation loss of the CNN in various cases for the recognition of Geez handwritten digits.

Table 3 describes the CNN configuration and parameters for the six cases. The models have varies numbers of convolutional and fully connected layers, as well as different layer organizations. The first fully connected layer contains 128 neurons and the second contains 20 neurons in all cases.

The first hidden layer in the first case presented in Figures 6 and 12 is the convolutional layer 1, which is used for feature extraction. It has 32 filters with a kernel size of 3×3 pixels, and it uses ReLU as an activation function. The next hidden layer is convolutional layer 2, which consists of 32

Table 1: Performance of the CNN for the six different cases for various hidden layers.

Case	Number of hidden layers	Batch size	*Minimum training accuracy*		*Minimum validation accuracy*		*Maximum training accuracy*		*Maximum validation accuracy*		Overall performance test accuracy (%)
			Epoch	Accuracy (%)	Epoch	Accuracy (%)	Epoch	Accuracy (%)	Epoch	Accuracy (%)	
1	9	32	1	88.15	1	91.75	28	99.23	18	95.82	95.65
2	14	32	1	85.01	1	89.00	28	98.74	20	94.99	94.71
3	13	32	1	85.96	1	89.63	26	98.63	20	95.28	94.98
4	10	32	1	88.88	1	91.89	30	99.36	27	95.64	95.42
5	8	32	1	87.41	1	89.90	29	99.77	27	94.84	94.42
6	12	32	1	88.77	1	91.94	30	98.44	12	96.15	96.21

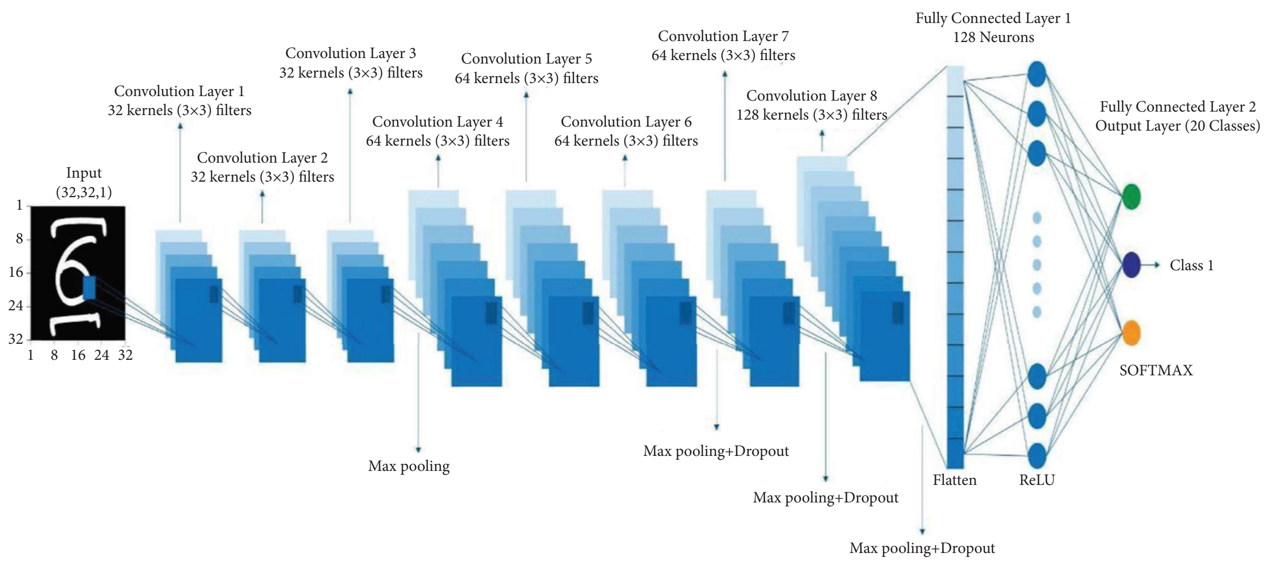

Figure 5: The convolutional neural network architecture of the proposed system.

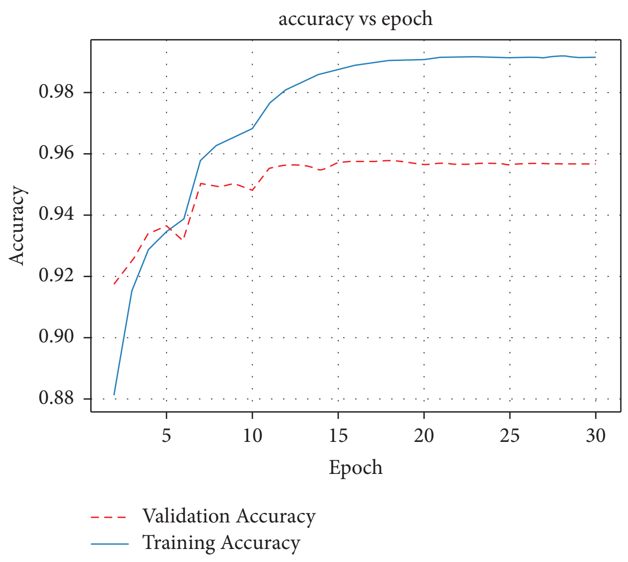

Figure 6: Observed accuracy for case 1.

filters with a kernel size of 3×3 pixels and ReLU. To minimize the spatial size of the output of a convolution layer, a pooling layer 1 is defined, with max-pooling and a pool size of 2×2 pixels. The next layers are two convolutional layers of a 64 filter with a kernel size of 3×3 pixel and the ReLU activation function is applied to the model. A max-pooling layer 2 is applied after the convolution layer4. Next to the pooling layer 2, a regularization layer dropout is used to reduce the

Table 2: Loss of the CNN for the six different cases for various hidden layers.

Case	Number of hidden layers	Batch size	*Minimum training loss*		*Minimum validation loss*		*Maximum training loss*		*Maximum validation loss*		Overall test loss
			Epoch	Loss	Epoch	Loss	Epoch	Loss	Epoch	Loss	
1	9	32	30	0.0232	8	0.2412	1	0.4515	1	0.3306	0.2946
2	14	32	28	0.0456	10	0.2571	1	0.5546	1	0.4149	0.2928
3	13	32	30	0.0423	14	0.2363	1	0.5214	1	0.4035	0.2908
4	10	32	27	0.0187	11	0.2371	1	0.4271	1	0.3266	0.3032
5	8	32	30	0.0068	8	0.3267	1	0.4705	28	0.4841	0.5504
6	12	32	24	0.0574	7	0.2077	1	0.4399	1	0.3213	0.2013

overfitting of the model by randomly eliminating 25% of the neurons in the layer. Convolution layer 5 with the channel size of 64 and filter size of 3×3 is applied after dropout. Convolutional layer 6 is the next hidden layer, which is made up of 128 filters with a kernel size of 3×3 pixels and ReLU. Max-pooling layer 3 with a dropout was applied after the convolution layer 6. A flattened layer is utilized to turn the 2D filter matrix into a 1D feature vector before entering the fully connected layers. After the flattened layer, the fully connected layer 1 is used, which comprises 128 neurons and ReLU. Finally, the fully connected layer 2 output layer, which determines the digits, has 20 neurons for 20 classes.

To output digits, the output layer has a SoftMax activation function. With a batch size of 32, the CNN is trained over 30 epochs. The performance has an overall test accuracy of 95.65%. The minimum validation accuracy is 91.75% at epoch 1, while the minimal training accuracy is 88.15% at epoch 1. At epoch 28, the highest training accuracy is 99.23%, whereas the highest validation accuracy is 95.82% at epoch 18. The overall model loss, in this case, is estimated to be around 0.2946. The training loss decreases exponentially when the iteration goes. The validation loss decreases from the pick value to the optimum value and then increases up to the 17th epoch. After the 19th epoch, the validation loss remains constant.

Figures 7 and 13 are defined for case2, where the first hidden layer is the convolutional layer 1, which is used for feature extraction. It has 32 filters with a kernel size of 3×3 pixels, and it uses ReLU as an activation function. The next hidden layer is convolutional layer 2, which consists of 32

Table 3: The CNN models for Geez handwritten digit recognition.

Model	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12
1	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Max-pooling 2×2@2@32	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Max-pooling 2×2@2@64 and dropout	Conv 3×3@1@64 and ReLU	Conv 3×3@1@128 and ReLU	Max-pooling 2×2@2@128 and dropout			
2	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Max-pooling 2×2@2@32	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Max-pooling 2×2@2@32	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Max-pooling 2×2@2@64 and dropout	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	3 additional layers such as pooling, conv, pooling
3	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Max-pooling 2×2@2@32	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Max-pooling 2×2@2@32 and dropout	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Max-pooling 2×2@2@64	Conv 3×3@1@64 and ReLU	Convolution layer with pooling and dropout layer
4	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Max-pooling 2×2@2@32	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Max-pooling 2×2@2@64 and dropout	Conv 3×3@1@128 and ReLU	Max-pooling 2×2@2@128 and dropout		
5	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Conv 3×3@1@64 and ReLU	Max-pooling 2×2@2@64 and dropout	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Conv 3×3@1@128 and ReLU	Max-pooling 2×2@2@128 and dropout				
6	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Conv 3×3@1@32 and ReLU	Max-pooling 2×2@2@32	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Conv 3×3@1@64 and ReLU	Max-pooling 2×2@2@64 and dropout	Conv 3×3@1@64 and ReLU	Max-pooling 2×2@2@64 and dropout	Conv 3×3@1@128 and ReLU	Max-pooling 2×2@2@128 and dropout

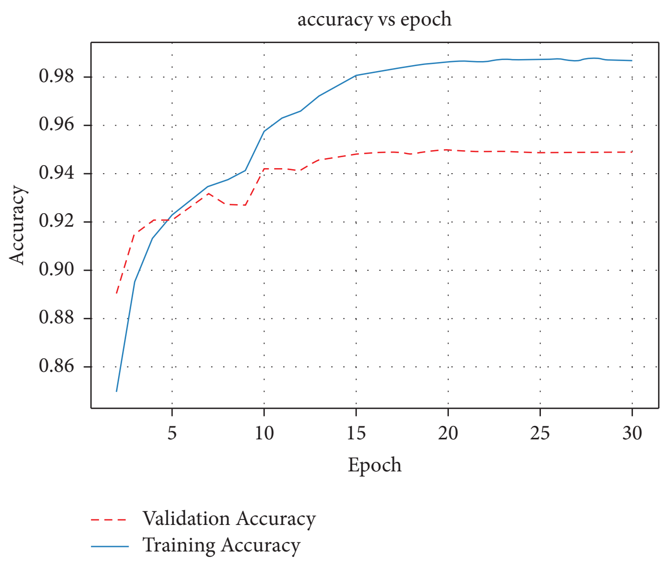

FIGURE 7: Observed accuracy for case 2.

filters with a kernel size of 3 × 3 pixels and ReLU. To minimize the spatial size of the output of a convolution layer, a pooling layer 1 is defined, with max-pooling and a pool size of 2 × 2 pixels. The next layers are two convolutional layers of a 32 filter with a kernel size of 3 × 3 pixel and the ReLU activation function is applied to the model. A max-pooling layer 2 is applied after the convolution layer4. The next two hidden layers are convolution layers which are made up of 64 filters with a kernel size of 3 × 3 pixels. Max pooling and dropout layers are applied after the convolution layers. The next two layers are convolution layers with a channel size of 64 followed by a max-pooling layer. The next hidden layer is convolution layer 9 with a 3 × 3 kernel size of 128 filters. A max-pooling layer with a dropout is applied after the convolution layer. Rectified Linear Units (ReLU) are used as an activation function in all convolution layers. The dimensions and hyperparameters used in this and the next cases are the same as those used in case 1. The overall performance test accuracy is found to be 94.71%. The minimal training and validation accuracy is determined at epoch 1. The training accuracy is 85.01%, and the validation accuracy is 89.00%. Epoch 28 has the highest training accuracy, while epoch 20 has the highest validation accuracy. The maximum accuracy for training and validation is 98.74% and 94.99%, respectively. The total model loss is estimated to be approximately 0.2928.

Two convolutions layers with a kernel size 3 × 3 which have 32 filters are taken one after the other in case 3, as shown in Figures 8 and 14, followed by a max-pooling layer. Two other convolution layers which have the same parameter from the first two layers are applied before the max-pooling layer and dropout layer. The next layers are three consecutive convolution layers which have 64 filter channels with a 3 × 3 kernel size and followed by a max-pooling layer. Before the flatten layer, two convolutional layers, max-pooling layer, and the dropout layer were applied. The two convolution layers have 64 and 128 kernel channels,

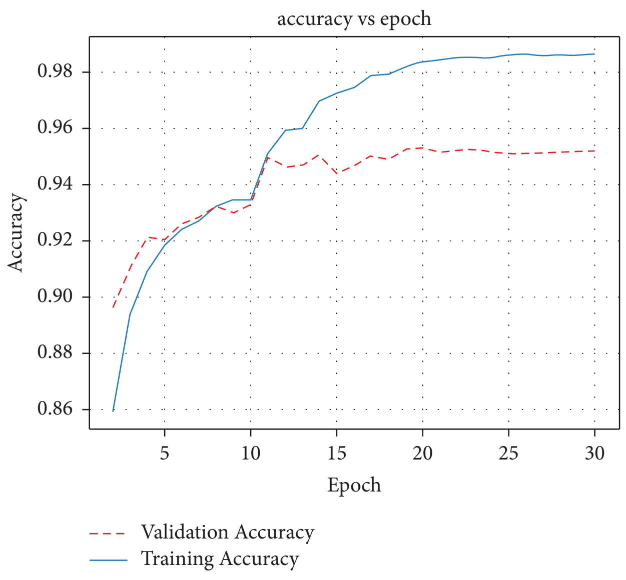

FIGURE 8: Observed accuracy for case 3.

respectively. Both layers have the same kernel size of 3 × 3. A flattened layer is followed by the two fully connected layers.

The overall performance test accuracy is found to be 94.98%. At epoch 1, the minimum training accuracy is 85.96%, whereas the minimum validation accuracy is 89.63%. The maximum training and validation accuracies are 98.63% and 95.28% found at epochs 26 and 20, respectively. The total model loss is found at approximately 0.2908.

For case 4, shown in Figures 9 and 15, three consecutive convolution layers are applied one after the other. The number of channel is 32 and the kernel size is 3 × 3. The max-pooling layer was applied after the three convolutional layers. The max-pooling layer is followed by three convolution layers which have 64 kernel channels and 3 × 3 kernel size which are followed by a max-pooling layer with a

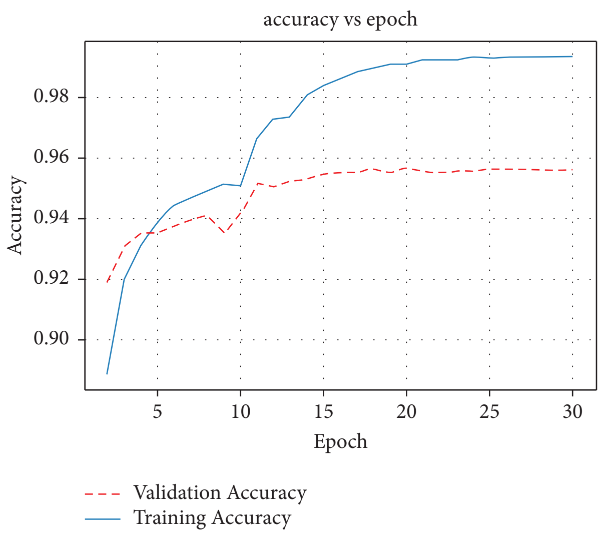

FIGURE 9: Observed accuracy for case 4.

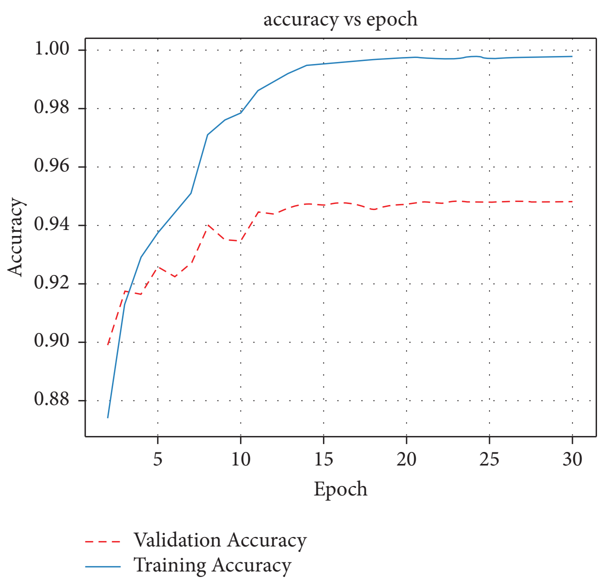

FIGURE 10: Observed accuracy for case 5.

dropout. The next layer is convolution layer 7 with a max-pooling layer and dropout. After a flattened layer, there are two fully connected layers with no dropout. The overall test accuracy in the performance is found 95.42%. At epoch 1, the minimum training and validation accuracies were found to be 88.88% and 91.89%, respectively. The maximum training accuracy is 99.36% is found at epoch 30, and the maximum validation accuracy is 95.64% is found at epoch 27. The total model loss is 0.3032.

Case 5 is shown in Figures 10 and 16, and for this case, three consecutive convolution layers are applied one after the other. The kernel channel is 32 and the kernel size is 3×3. The max-pooling layer was applied after the three convolutional layers. Next to the pooling layer, a regularization layer dropout is used to reduce overfitting by randomly eliminating 20% of the neurons in the layer. The next layers are three convolution layers followed by a max-pooling layer and a dropout layer. The two fully connected layers are followed by a flattened layer.

The overall performance test accuracy was found to be 94.42%. At epoch 1, the minimum training accuracy is 87.41%, while the minimum validation accuracy is 89.90%. Epoch 29 has the highest training accuracy, while epoch 27 has the highest validation accuracy. The maximum accuracy for training and validation is 99.77% and 94.84%, respectively. The total test loss of the model is 0.5504. The validation loss of the model increase when the iteration goes. It shows the model became overfit to the training data. The maximum model loss is occurred in this case from all the six cases. Also, the minimum model accuracy among all cases occurred in case 5. It shows that overfitted models give a high model loss and low accuracy for a new test dataset.

Finally, in Case 6 (Figures 11 and 17), three convolutions are taken one after the other, followed by a pooling layer. The three convolution layers have 32 kernel channels. Three convolution layers with a kernel size 64 are next, followed by a max-pooling layer. Next to the pooling layer 2, a regularization layer dropout is applied to reduce overfitting by randomly eliminating 20% of the neurons in the layer. Convolutional layer 7 which has 64 kernel size is the next hidden layer, followed by a max-pooling layer and a dropout layer. The next layer is convolution layer 8 with 128 number of channels and kernel filter size of 3×3. All convolution layers have the same filter size. Max-pooling layer 4 with a dropout was applied after the convolution layer 8. The flatten layer, followed by two fully connected layers, is applied. The overall performance test accuracy was found to be 96.21%. At epoch 1, the minimum training and validation accuracies were found to be 88.77% and 91.94%, respectively. Epoch 30 has the highest training accuracy, while epoch 12 has the highest validation accuracy. The maximal training and validation accuracy is 98.44% and 96.15%, respectively. The total model loss is found approximately 0.2013. The training

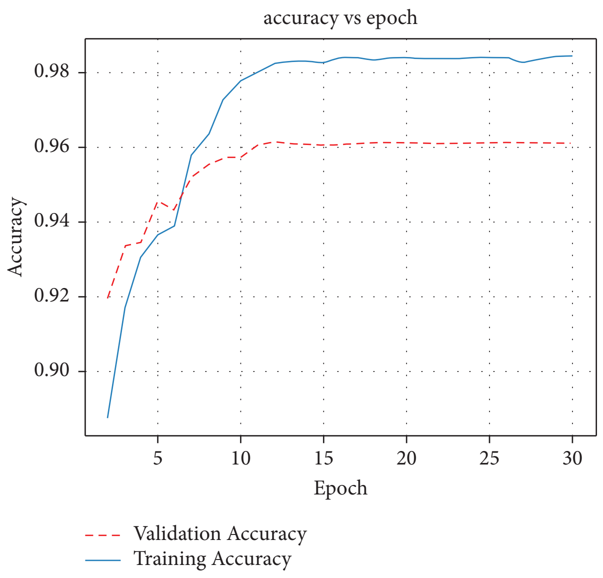

FIGURE 11: Observed accuracy for case 6.

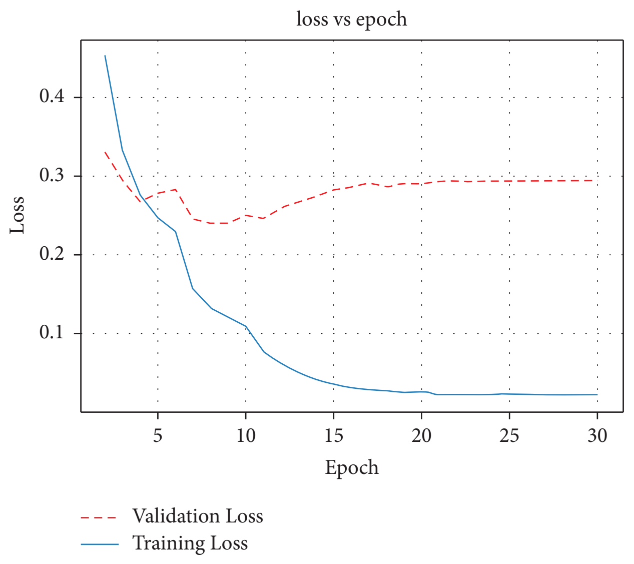

Figure 12: Observed loss for case 1.

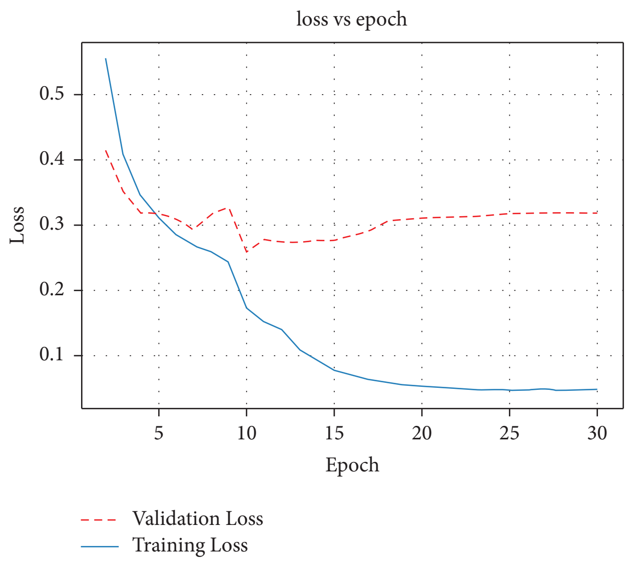

Figure 13: Observed loss for case 2.

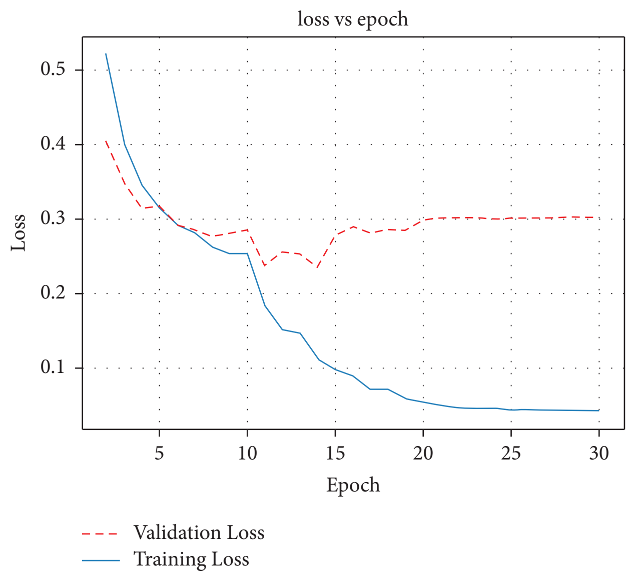

Figure 14: Observed loss for case 3.

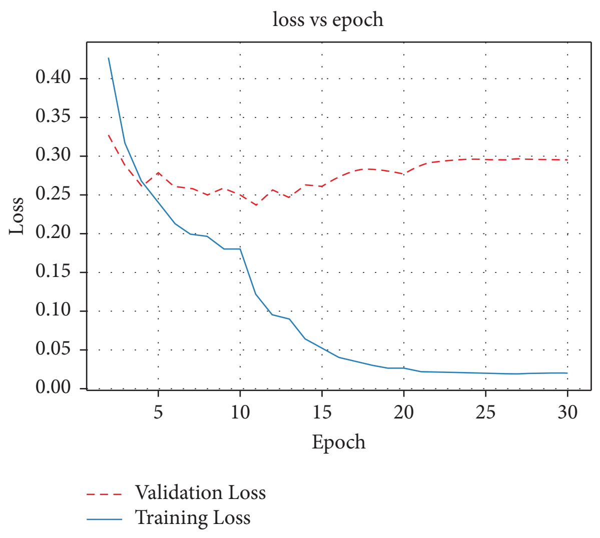

Figure 15: Observed loss for case 4.

loss decreases when the number of epoch goes, but the validation loss fluctuate for 10 epochs and then remain constant for the remaining number of epochs.

By varying the hidden layers, the changes inaccuracies for handwritten digits were observed over 30 epochs in the experiment. Accuracy curves for the six cases for each parameter were generated using a handwritten Geez digit dataset. The six cases behave differently due to the different combinations of hidden layers. The maximum and minimum accuracies for several hidden layer variations were recorded using a batch size of 32. As shown in Figure 18, the highest test accuracy in performance was found to be 96.21% for 30 epochs in case 6 among all the observations (Conv1, Conv2, Conv3, pool1, Conv4, Conv5, Conv6, pool2 with dropout, Conv7, pool3 with dropout, Conv8, pool4 with dropout, flatten layer, 2 fully connected layers).

This type of greater accuracy will work in Geez handwritten digit recognition to help the machine execute more efficiently. In case 5, however, the lowest accuracy among all observations in the performance was discovered to be 94.42% (Conv1, Conv2, Conv3, pool1, Conv4, Conv5, Conv6, pool2, flatten layer, and 2 fully connected layers). Furthermore, the total highest model loss in case 5 is 0.5504, while the total lowest model loss in case 6 with dropout is around 0.2013 (Figure 19). With this minimal loss, the CNN will be able to achieve greater image quality and noise processing. From the observed result, the study chooses the best model from six cases that have highest model test accuracy and lowest test loss. So, case 6 model with highest accuracy of 96.21% and lowest loss of 0.2013 is the proposed model for this research work.

The previous work on Geez handwritten digit recognition is done by the author of [3] who achieved 89.88%

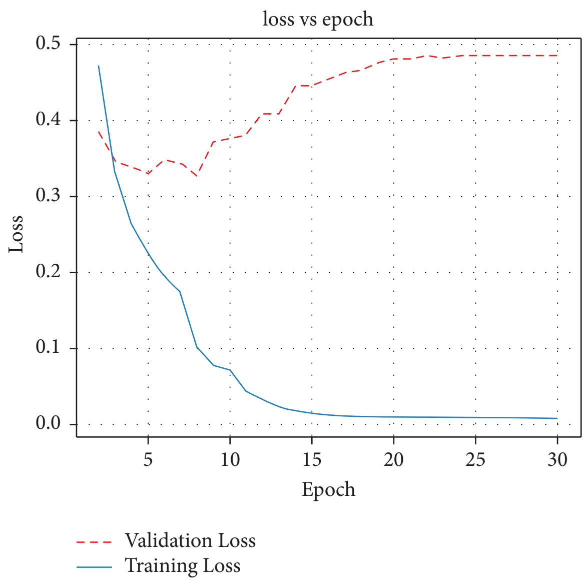

Figure 16: Observed loss for case 5.

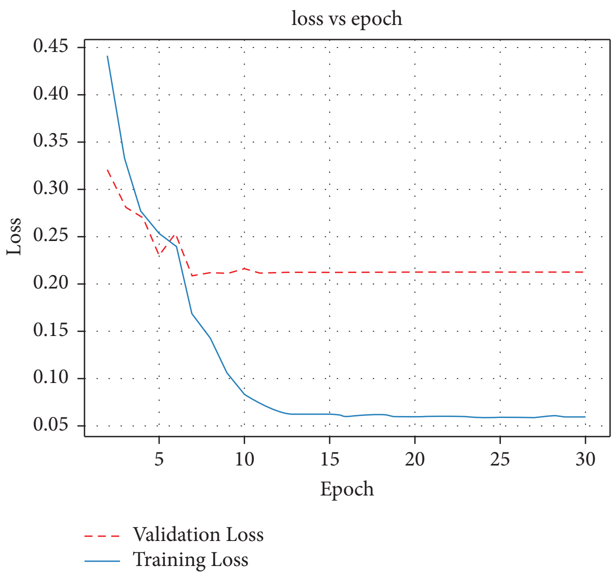

Figure 17: Observed loss for case 6.

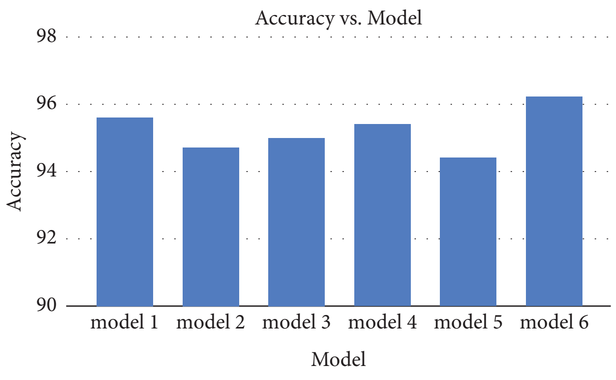

Figure 18: Accuracy of the different models.

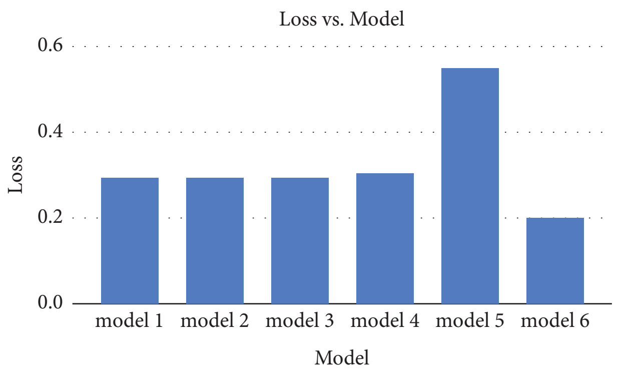

Figure 19: Loss of the different models.

accuracy using an ANN model. This study evaluates CNN models with different layers with different hyperparameters. Compared with the previous work, the study improve the accuracy of the recognition from 89.88% to 96.21% by using CNN, increasing the dataset size, and enhancing the quality of the image by using pre-processing techniques on the dataset.

7. Conclusion and Future Scope

In this research work, convolutional neural networks was used to recognize Geez handwritten digits with 20-digit classes. CNNs are the current state-of-the-art algorithm for classifying image data and are widely used. On a prepared form for data collection, a large number of Geez handwritten digits were collected from individual handwriting. The handwritten documents are scanned and preprocessed to get 32×32-pixel digit images. The study offered a new public dataset for the Geez handwritten digit dataset, which is open to all researchers. CNN architecture was used from the deep learning approaches to develop an Geez handwritten digit recognition system. A lot of trial and error neural network configuration tuning mechanisms were used to get the best fit model of CNN-based architecture. In comparison to earlier research works on Geez handwritten digit recognition, the study able to achieve higher recognition accuracy using the developed CNN model. The proposed model achieved an accuracy of 96.21% and a model loss of 0.2013.

Regardless of the fact that much work has been done in the English language to recognize handwritten digits, only a small amount of work has been done in the Amharic language. Due to a lack of research work on the area, there is a big challenge to get datasets for the Amharic language. The collected data amount is enough to train the model, but it is not a large dataset, and the students dominate the respondent of the data gathering. Most of the respondent is student, so the model is performed well for the students and for other individual group the model does not perform well like the students. The dataset does not include the historical document and manuscript images. The collected data are only from individuals not including other sources. In this research, a dataset was developed that can be used by other researchers in the future. In the future, the dataset will also have historical data as the dataset for the model, and the current work only supports a single handwritten Ge’ez digit, but in the future, add the support for multi-digit.

Data Availability

The data used to support the findings of this study are available at https://drive.google.com/file/d/1abJWvSYSyw8mLQ5Blg_lYAJng1K3LtGS/view?usp=sharing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The study was funded by the university.

References

[1] F. A. Demilew and B. Sekeroglu, “Ancient Geez script recognition using deep learning,” *SN Applied Sciences*, vol. 1, no. 11, pp. 1315–1317, 2019.

[2] M. S. Gondere, L. Schmidt-thieme, A. S. Boltena, and H. S. Jomaa, “Handwritten Amharic Character Recognition Using a Convolutional Neural Network,” 2019, https://arxiv.org/abs/1909.12943#:%7E:text=This%20research%20work%20designs%20for,was%20applied%20for%20machine%20learning.

[3] E. G. Beyene, “Handwritten and machine printed ocr for geez numbers using artificial neural network,” 2019, https://arxiv.org/abs/1911.06845.

[4] E. Granell, E. Chammas, L. Likforman-Sulem, C. D. Martínez-Hinarejos, C. Mokbel, and B. I. Cirstea, “Transcription of Spanish historical handwritten documents with deep neural networks,” *Journal of Imaging*, vol. 4, no. 1, pp. 15–22, 2018.

[5] H. Kusetogullari, A. Yavariabdi, J. Hall, and N. Lavesson, “Digitnet: a deep handwritten digit detection and recognition method using a new historical handwritten digit dataset,” *Big Data Research*, vol. 23, Article ID 100182, 2021.

[6] O. Elitez, “Handwritten digit string segmentation and recognition using deep learning,” Master’s Thesis, Middle East Technical University, Ankara, Turkey, 2015.

[7] F. C. Ribas, L. S. Oliveira, A. S. Britto Jr, and R. Sabourin, “Handwritten digit segmentation: a comparative study,” *International Journal on Document Analysis and Recognition*, vol. 16, no. 2, pp. 127–137, 2013.

[8] R. Saabni, “Recognizing handwritten single digits and digit strings using deep architecture of neural networks,” in *Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition*, pp. 1–6, IEEE, Lodz, Poland, 2016.

[9] Z. Shi and F. Date, “Detecting date regions on handwritten document images based on positional expectancy,” Master’s Thesis, University of Groningen, Groningen, Netherlands, 2016.

[10] D. Ciresan, “Avoiding segmentation in multi-digit numeral string recognition by combining single and two-digit classifiers trained without negative examples,” in *Proceedings of the International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 225–230, IEEE, Timisoara, Romaniapp, 2008.

[11] A. Dutt and D. Aashi, “Handwritten digit recognition using deep learning,” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 6, no. 7, pp. 990–997, 2017.

[12] F. Siddique, S. Sakib, and M. A. B. Siddique, “Recognition of handwritten digit using convolutional neural network in python with tensorflow and comparison of performance for various hidden layers,” in *Proceedings of the 2019 5th International Conference on Advances In Electrical Engineering (ICAEE)*, pp. 541–546, IEEE, Dhaka, Bangladesh, 2019 September.

[13] I. S. Krizhevsky and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[14] Y. LeCun, “LeNet-5, convolutional neural networks,” 2015, https://yann.lecun.com/exdb/lenet.

[15] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[16] S. Chen, R. Almamlook, Y. Gu, and L. Wells, “Offline handwritten digits recognition using machine learning,” in *Proceedings of the International Conference on Industrial Engineering and Operations Management*, pp. 274–286, Washington, DC, USA, September 2018.

[17] M. A. Hossain and M. M. Ali, “Recognition of handwritten digit using convolutional neural network (CNN),” *Global Journal of Computer Science and Technology*, vol. 19, no. 2, pp. 27–33, 2019.

[18] S. Ali, Z. Shaukat, M. Azeem, Z. Sakhawat, T. Mahmood, and K. Ur Rehman, “An efficient and improved scheme for handwritten digit recognition based on convolutional neural network,” *SN Applied Sciences*, vol. 1, no. 9, pp. 1125–1129, 2019.