IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 31, 2019, accepted December 6, 2019, date of publication December 16, 2019, date of current version January 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2960161

A New Hybrid Convolutional Neural Network and eXtreme Gradient Boosting Classifier for Recognizing Handwritten Ethiopian Characters

HALEFOM TEKLE WELDEGEBRIEL[1], HAN LIU[2], (Member, IEEE), ANWAR UL HAQ[1], EMMANUEL BUGINGO[1], AND DEFU ZHANG[1], (Member, IEEE)

[1]School of Informatics, Xiamen University, Xiamen 361005, China
[2]School of Computer Science and Informatics, Cardiff University, Cardiff CF24 3AA, U.K.

Corresponding author: Defu Zhang (dfzhang@xmu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672439, and in part by the China Government Scholarship Council under Grant 2016MOC213.

ABSTRACT Handwritten character recognition has been profoundly studied for many years in the field of pattern recognition. Due to its vast practical applications and financial implications, the handwritten character recognition is still an important research area. In this research, a Handwritten Ethiopian Character Recognition (HECR) dataset is prepared to train a model. Images in the HECR dataset were organized with more than one color pen RGB main spaces that are size normalized to 28 × 28 pixels. The dataset is a combination of scripts (Fidel in Ethiopia), numerical representations, punctuations, tonal symbols, combining symbols, and special characters. These scripts have been used to write ancient histories, science, and arts of Ethiopia and Eritrea. In this study, a hybrid model of two super classifiers: Convolutional Neural Network (CNN), as well as eXtreme Gradient Boosting (XGBoost), are proposed for classification. In this integrated model, CNN works as a trainable automatic feature extractor from the raw images and XGBoost takes the extracted features as an input for recognition and classification. The output error rates of the hybrid model and CNN with a fully connected layer are compared. A 0.4630 and 0.1612 error rates were achieved in classifying the handwritten testing dataset images, respectively. The XGBoost as a classifier gave better results than the traditional fully connected layer.

INDEX TERMS Convolutional neural network (CNN), eXtreme gradient boosting (XGBoost), feature extraction, pattern recognition.

NOTATIONS

A concise reference describing the notation used throughout this paper is as follows:

x_i	The i^{th} example (input) from the dataset
y_i	The target value associated with $x^{(i)}$ for supervised learning
$\hat{y}$	Predicted value
D	Dataset
$h(x)$	Function
F_1	Objective function
$\mathbb{R}$	Set of real numbers
θ	Weight
b	Bias
α	Learning rate
W	Input volume size (input image width)
F	Filter Size
S	Stride
P	Zero padding
$H(x)$	Hessian matrix
$g(x)$	Gradient vector
$A(x)$	Jacobian matrix
$\Omega(h)$	Regularization complexity
R	Functional space (all possible set of classification and regression trees)
Υ	Gamma
L	Number of leaves in the XGBoost tree
λ	L_2 regularization
ω	Vector score in XGBoost leaves
σ	Activation function
C	Convolution layer
S	Pooling layer

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

I. INTRODUCTION

In the field of pattern recognition, handwritten character recognition has been widely studied for many years. Furthermore, the handwritten recognition is an important research area due to its vast practical applications and financial implications. Many industries are in need of a decent recognition rate with the highest reliability. Interestingly, industries in works related to real-life problems are more interested in reliability rather than recognition accuracy [1]. Recognition, interpretation, and identification of documents are three major known handwritten document manipulation. Handwriting recognition is the process of transforming a language represented in its spatial form of graphical symbols into its symbolic representation. Handwriting interpretation refers to the task of determining the meaning of a body of handwriting. Handwriting identification is the process of identifying the author of a handwriting sample from a set of writers, assuming that each person's handwriting is distinctive [2], [3].

Several automatic systems of offline handwriting recognition are already available in the marketplace. However, these systems provide solutions mainly for major world scripts such as English, Japanese, Chinese, and Arabic. Moreover, the recognition problems of these scripts are still not solved entirely. On the other hand, there are many scripts used in different parts of the world for which no automatic system for recognizing the handwritten character images, exists. Furthermore, due to massive variability in handwriting styles, the state-of-the-art handwriting recognition technologies often fail to provide satisfactory performance on various types of handwriting samples.

In the recent past, handwriting recognition technologies have progressed rapidly and significant improvements have been achieved by using different algorithms, such as Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Artificial Neural Networks (ANN) [4]. A gradient boosting algorithm was developed for very high predictive competence. However, its espousal is very limited because the algorithm requires one decision tree at a time in minimizing the errors of all previous trees in the model. Therefore, it takes longer to train even the simplest models. To overcome this problem a new algorithm called XGBoost was discovered. In the XGBoost, individual trees are created using multiple cores and data is organized for minimizing the lookup time, which reduces the training time of models and also improves classification accuracy. The most important factor behind the success of XGBoost is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory limited settings. Several systemic and algorithmic optimizations are key factors for the scalability of XGBoost [5].

On the other hand, CNN is a special kind of ANN designed to explore the geometry of images in order to accommodate the prior knowledge of the model. The CNN is built using a hierarchical architecture for learning deeper features from the image dataset. Scalability is one of the main advantages presented by these models. Nonetheless, CNN has some drawbacks. In contrast to XGBoost, since the loss surface is non-convex, a global minimal is not assured in this case. Furthermore, the last layer of CNN is a traditional fully connected linear layer classifier and less efficient than an XGBoost. Therefore, to overcome the drawbacks of CNN and XGBoost, a hybrid technique of CNN and XGBoost is used in this paper. This combination helped to address the limitations of both techniques while combining the best characteristic of each to enhance accuracy. First, CNN is trained using backpropagation in order to extract features. Second, the extracted features are given as an input to the XGBoost for classification.

Among the many languages in Ethiopia Geez, Tigrigna and Amharic are some of the Semitic languages that have morphologically diverse scripts and numbers of their own. These scripts and numbers are almost the same except one additional base script with its orders and some special scripts exist in Tigrigna (Table 1). They have their own consonants (base scripts) and vowels (orders). An abugida is a technique used to write these languages. Geez has been spoken in Ethiopia since the Axumite kingdom. However, it is currently used as the only language of liturgy in Ethiopian and Eritrean Orthodox Tewahdo Churches. Many of the ancient histories, science, and arts of Ethiopia and Eritrea are handwritten documents. To preserve these documents, modern commercial Optical Character Recognition (OCR) software is required. The present study mainly focuses on the recognition part of OCR. Many researchers have performed handwritten character recognition based on their country's characters. Like any other country, the main challenges of research in Ethiopian indigenous scripts recognition are the use of a large number of scripts in the writing, and the existence of a large number of visually identical scripts.

As Ethiopian handwritten character recognition has not been studied yet, the main contributions of this study are the following: 1) 502 Ethiopian scripts were collected and ordered in their sequence; 2) as there was no existing offline dataset for Ethiopian scripts, a new dataset is prepared in 28×28 pixels that are manually cropped; 3) for the first time, a handwritten dataset that was prepared with more than one color pen is used; and 4) a combined model of CNN and XGBoost is proposed for Ethiopian handwritten scripts recognition.

The rest of this paper is organized as follows: Section II reviews the related work in the field of handwritten character recognition. Section III describes the deep learning concepts and algorithms. Section IV reviews the material and method used. Section V elaborates experimental results and analysis in detail. Section VI presents some conclusions and potential future research directions.

II. RELATED WORKS

Handwritten character based on the most frequently used 520 types of Hangul characters in Korea is reported by Park and Lee [6] using a framework of stochastic models,

TABLE 1. Most widely used ethioplan scripts.

	Base	Orders	Base	Orders
Basic Scripts	ሀ	ሁ ሂ ሃ ሄ ህ ሆ	ከ	ኩ ኪ ካ ኬ ክ ኮ
	ለ	ሉ ሊ ላ ሌ ል ሎ	ኸ	ኹ ኺ ኻ ኼ ኽ ኾ
	ሐ	ሑ ሒ ሓ ሔ ሕ ሖ	ወ	ዉ ዊ ዋ ዌ ው ዎ
	መ	ሙ ሚ ማ ሜ ም ሞ	ዐ	ዑ ዒ ዓ ዔ ዕ ዖ
	ሠ	ሡ ሢ ሣ ሤ ሥ ሦ	ዘ	ዙ ዚ ዛ ዜ ዝ ዞ ዟ
	ረ	ሩ ሪ ራ ሬ ር ሮ	ዠ	ዡ ዢ ዣ ዤ ዥ ዦ
	ሰ	ሱ ሲ ሳ ሴ ስ ሶ	የ	ዩ ዪ ያ ዬ ይ ዮ
	ሸ	ሹ ሺ ሻ ሼ ሽ ሾ	ደ	ዱ ዲ ዳ ዴ ድ ዶ
	ቀ	ቁ ቂ ቃ ቄ ቅ ቆ	ጀ	ጁ ጂ ጃ ጄ ጅ ጆ
	ቐ	ቑ ቒ ቓ ቔ ቕ ቖ	ገ	ጉ ጊ ጋ ጌ ግ ጎ
	በ	ቡ ቢ ባ ቤ ብ ቦ	ጠ	ጡ ጢ ጣ ጤ ጥ ጦ
	ቨ	ቩ ቪ ቫ ቬ ቭ ቮ	ጨ	ጩ ጪ ጫ ጬ ጭ ጮ
	ተ	ቱ ቲ ታ ቴ ት ቶ	ጰ	ጱ ጲ ጳ ጴ ጵ ጶ
	ቸ	ቹ ቺ ቻ ቼ ች ቾ	ጸ	ጹ ጺ ጻ ጼ ጽ ጾ
	ኀ	ኁ ኂ ኃ ኄ ኅ ኆ	ፀ	ፁ ፂ ፃ ፄ ፅ ፆ
	ነ	ኑ ኒ ና ኔ ን ኖ	ፈ	ፉ ፊ ፋ ፌ ፍ ፎ
	ኘ	ኙ ኚ ኛ ኜ ኝ ኞ	ፐ	ፑ ፒ ፓ ፔ ፕ ፖ
	አ	ኡ ኢ ኣ ኤ እ ኦ		
Special Scripts		ኰ ኲ ኳ ኵ ዀ ዂ ዃ ዅ	ቈ ቊ ቋ ቍ ኈ ኊ ኋ ኍ	ቘ ቚ ቛ ቝ ጐ ጒ ጓ ጕ
		ሏ ሗ ሟ ሧ ሯ ሷ ሿ ቋ ቛ ቧ ቯ ቷ ቿ ኋ ኗ ኟ ኧ ኳ ዃ ዟ ዧ ዷ ጇ ጓ ጧ ጯ ጷ ጿ ፏ ፗ		
Numbers		፩ ፪ ፫ ፬ ፭ ፮ ፯ ፰ ፱ ፲ ፳ ፴ ፵ ፶ ፷ ፸ ፹ ፺ ፻ ፼		

the first-order hidden Markov models (HMMs). A deep CNN with 5 convolutional and 3 fully connected layers were also used to evaluate handwritten letter recognition [7]. Keysers *et al.* [8] used Google online handwriting recognition systems for multiple-language such as Chinese, Japanese, and Korean based on ANN. Pradeep *et al.* [9] applied ANN in handwritten character recognition of 38 English alphabets and numbers. The experiment by Katiyar *et al.* [10] used a well-known standard database acquired from CEDAR that contains images of alphabetic and numeric characters and for evaluating accuracy, SVM was used.

Arabic handwritten numeric character recognition using deep learning neural networks is addressed by Tushar and Ashiquzzaman [11]. In this case, Multi-Layer Perceptron (MLP) and CNN were used for comparison. Younis [12] also applied a deep CNN on Arabic handwritten character recognition. In knowing the best methodology in recognizing handwritten Marathi characters, SVM and Feed-Forward ANN were compared by Kamble and Hegadi [13]. The rectangular histogram-oriented gradient was used to extract the features of the characters, while SVM was applied as a classifier. Suganthi and Dineshkumar [14] used a feed-forward ANN to recognize Sanskrit characters. CASIA is an offline dataset for Chinese characters. Many handwritten character recognition research works have been reported using this dataset as can be found in a research paper by Chen [15] in which CNN was applied to the dataset. Using different architecture, Zhang [16] also used a deep CNN to recognize Chinese handwritten characters on the same dataset.

Although several studies have reported on Ethiopian-Amharic OCR, they did not include all the Ethiopian scripts. Zewidie [17] included only 15 Ethiopian scripts using SVM. On the other hand, Birhanu and Sethuraman [18] used an ANN approach to develop OCR for real-life Amharic documents. Meanwhile, Jawahar and Meshesha [19] worked on Amharic OCR using the SVM classifier on top of the existing Indian language classifier. The latest paper by Weldegebriel *et al.* [20] on Ethiopian Geez script using CNN and a feedforward MLP has followed two steps: First, MLP compared four widely used Ethiopic typefaces such as Ethiopian Jiret, Ebrima, Nyala and Abyssinica SIL. Images of 26 basic Ge'ez scripts written from those typefaces were given to the system as input for training. Scanned documents were given again as an input to check which of those typeface scripts are recognized and classified well. As a result, scripts written in Ethiopian Jiret typeface are recognized very well. Second, using Ethiopian Jiret typeface many images of strings are generated using a standard python codec registry. These images are used as an input to the CNN and MLP for comparison.

III. DEEP LEARNING CONCEPTS

In a time when a large number of datasets and supercomputers were difficult to acquire, it was not possible to think about deep learning. Currently, there are enormous amounts of structured and unstructured data on the net. Improvements in computer hardware and network structure have enabled the training of truly deep CNNs only recently [21]. In order to manipulate the existing resources, some concepts and

algorithms have been developed. To mention some of them, deep learning is briefly introduced in section A. Then, supervised learning and soft-max are described in-depth under section B and C, respectively. Finally, the feedforward network is presented in section D.

A. DEEP LEARNING

Deep learning is a branch of machine learning in artificial intelligence (AI) that has networks capable of learning patterns from a given data. It is a machine learning paradigm that focuses on learning deep hierarchical models of data. Deep learning is most easily explained in contrast to more shallow learning methods. An archetypical shallow learning method might be a feedforward NN with an input layer, a single hidden layer and an output layer trained with backpropagation on a classification task [22]. It is known that a regular NN receives an input (a single vector), and transforms it through a series of hidden layers. Each hidden layer is made up of a set of different numbers of neuron units, where each neuron unit is fully connected to all neurons in the previous layer. The neurons in a single layer function are completely independent and do not share any connections. The last fully-connected layer is known as the output layer. In a classification task, the output layer represents the class scores (posterior probability for each class). Regular neural networks do not scale well for large image sizes.

CNN takes advantage of the fact that unlike regular NN, the neurons of convolutional network layers are arranged in 3 dimensions which are width, height, and depth. The term depth here refers to the third dimension of an activation volume (channel), not to the depth of a full NN, which can refer to the total number of layers in a network. The neurons in a layer are connected to a small region of the layer in the previous layer instead of all of the neurons in a fully-connected manner. As a result, all the neurons have the same connection weights, and the last fully connected output layer would have a dimension of $1 \times 1 \times c$ (c refers to the number of classes) as a full image is reduced to a single vector of class scores. As described above, a convolutional network is a sequence of different layers and every layer of a convolutional network transforms a given input volume of activations to another output through a differentiable function.

The main types of layers that build a CNN model are the convolution layer, rectifying linear unit, dropout, pooling layer, and fully connected layer [23]. These layers are going to be thoroughly explained in the CNN classifier section. In machine learning, there are many learning algorithms. Supervised, unsupervised, and reinforcement are the most commonly known learning mechanisms. Based on their applications, algorithms classified under these learnings can be divided into continuous and categorical groups.

B. SUPERVISED LEARNING

Supervised learning is the most popular machine learning technique that many computer vision researchers have been conducting their research using it. Supervised classification of an image is done when the classification categories are defined. The data is divided into training and testing sets, but part of the dataset is also taken as a validation set from the training set after the testing set is separated from the dataset. The labeled data will be fed to the machine learning algorithm for training. The algorithm is trained on the labeled dataset and gives the desired output (the pre-defined categories). During the testing phase, the algorithm is fed with data that has never been seen before and classifies them into categories based on the knowledge it got during the training phase [24], [25]. According to Andrew [26], in order to perform supervised learning, the representation of functions or hypotheses (h) must be decided first. As an initial choice, let y is approximated as a linear function of x:

$$h(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \tag{1}$$

considering the intercept term $x_0 = 1$, so that:

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x \tag{2}$$

To formalize this, a nonlinear cost function is defined to measure, for each value of the θ how close $h(x^{(i)})$ are to the corresponding $y^{(i)}$:

$$F_1(\theta) = \frac{1}{2} \sum_{i=0}^{m} (h(x^{(i)}) - y^{(i)})^2 \tag{3}$$

Weight θ must be chosen so as to minimize the function $F_1(\theta)$. To do so, a search algorithm is used that starts with some initial guess values for θ, and that repeatedly changes θ to make the value of $F_1(\theta)$ smaller, until it converges to a value of θ that minimizes $F_1(\theta)$. Specifically, consider the gradient descent algorithm, which starts with some initial values of θ, and a learning rate α, then repeatedly performs the update:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} F_1(\theta) \tag{4}$$

In order to implement this algorithm, the partial derivative term on the right-hand side must be performed as follows. In the first case, only one training example (x, y) is considered. The equation on the left side is from (4) and the right side is from (3). Neglecting the sum in the definition of $F_1(\theta)$, the final result of the derivation is:

$$\begin{aligned} \frac{\partial}{\partial \theta_j} F_1(\theta) &= \frac{\partial}{\partial \theta_j} \times \frac{1}{2}(h(x) - y)^2 \\ &= 2 \times \frac{1}{2}(h(x) - y) \times \frac{\partial}{\partial \theta_j}(h(x) - y) \\ &= (h(x) - y) \times \frac{\partial}{\partial \theta_j}(\sum_{i=0}^{n} \theta_i x_i - y) \\ &= (h(x) - y) x_j \end{aligned} \tag{5}$$

Substituting (5) in (4) gives

$$\theta_j := \theta_j + \alpha (y^{(i)} - h(x^{(i)})) x_j^{(i)} \tag{6}$$

Due to many existing optimization techniques in machine learning, many researchers use different optimizers to minimize their losses and increase accuracy. The ultimate goal of

this mathematical calculation is to get a good model that can generalize for unseen data. The stochastic gradient descent algorithm (also called incremental gradient descent) is used in this work. Stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at. Often, stochastic gradient descent gets θ close to the minimum much faster than batch gradient descent (However, note that, it may never converge to the minimum, and the parameter θ will keep oscillating around the minimum of $F_1(\theta)$, but in practice, most of the values near the minimum will be reasonably good approximations to the true minimum). The stochastic gradient descent algorithm is shown below.

for every j {
  for i = 1 *to m* {
    $\theta_j := \theta_j + \alpha(y^{(i)} - h(x^{(i)}))x_j^{(i)}$
  }
}

When a training set is large, stochastic gradient descent is often preferred over batch gradient descent as batch gradient descent tries to scan through the entire training set before taking a single step which is a costly operation if the training sample (n) is very large [26], [27].

Models often involve some unknown parameters. The unknown parameters of a model have to be assessed before solving the model. Parameters can be determined using the model and the observed data from the real problem. Parameter estimation problems take various forms, i.e. nonlinear optimization, nonlinear equations or nonlinear least squares which can be solved by numerical methods. There are also many numerical methods, which are appropriate for a specific variety of problems. Newton's method and its variants are some of the methods enabling us in solving parameter estimation problems of small to medium scale. Newton's method may fail if the Hessian (Jacobian) matrices are singular or indefinite (singular). Newton's method with trust-region has been used to avoid such a problem. Equation (3) is nonlinear least square and has been described well mathematically by Kabir [28]. Given input-output pairs of (t_i, y_i), $i = 1, \ldots, m$, it is possible to find a vector $x \in \mathbb{R}^n$ that gives the best fit in the least square sense to model a function $h(t, x)$, where $h : \mathbb{R}^n \to \mathbb{R}^m$.

The components of the residual $F_1 : \mathbb{R}^n \to \mathbb{R}^m$ are defined as $F_1(x) = y - h(t_i, x), \forall_i = 1, \ldots, m$.

The aim is to compute the vector x which minimizes the difference between y_i and $h(t_i, x)$ for all $i = 1, \ldots, m$. This can be considered as a problem where the sum of squares of residual components is to be minimized.

$$\min_x \phi(x) = \min_x \frac{1}{2} F_1(x)^T F_1(x) \tag{7}$$

The value $\frac{1}{2}$ is taken for convenience and it has no effect on the optimal value of x. in practice, m is significantly larger than n and the experimental errors yield $y_i \neq h(t_i, x)$ even with the best possible x. Applying Newton's method to (7) a Newton step s_k for the current iterate x_k can be obtained from solving

$$\underbrace{\nabla^2\phi(x_k)}_{H(x_k)} s_k = \underbrace{-\nabla\phi(x_k)}_{g(x_k)} \text{ for } s_k \tag{8}$$

The Hessian matrix $H(x_k)$ and the gradient vector $g(x_k)$ is given by

$$\begin{aligned} H(x_k) &= A(x_k)^T A(x_k) + \underbrace{\sum_{i=0}^{m} \nabla^2 F_i(x_k) F_i(x)}_{s_k} \\ &= A(x_k)^T A(x_k) + s_k \end{aligned} \tag{9}$$

$$\begin{aligned} g(x_k) &= \sum_{i=0}^{m} \nabla F_i(x_k) F_i(x) \\ &= \underbrace{\nabla F_1(x_k)^T}_{A(x_k)} F_1(x_k) \\ &= A(x_k)^T F_1(x_k) \end{aligned} \tag{10}$$

The s_k in (9) is usually expensive to compute. To avoid much computing cost s_k is omitted and obtains

$$H(x_k) \approx A(x_k)^T A(x_k).$$

Hence (8) can be rewritten as

$$A(x_k)^T A(x_k) s_k = -A(x_k)^T F_1(x_k),$$

which is the system of normal equations for the Gauss-Newton equations

$$A(x_k) s_k = -F_1(x_k) \tag{11}$$

To avoid singularity in $A(x_k)$ the regularization described by Kabir [28] in section 4.3 is used for (11) which gives the Levenberg-Marquardt equations

$$\begin{pmatrix} A(x_k) \\ \sqrt{yk} I \end{pmatrix} s = - \begin{pmatrix} F_1(x_k) \\ 0 \end{pmatrix} \tag{12}$$

where $\sqrt{yk}$ is a positive parameter chosen by some strategy in order to avoid an inappropriate condition of the linear least-squares sub-problem 11, and I is the initial approximation of the Hessian unit matrix. If the nonlinear least square (residual) $\|F_1(x)\|$ is very large, then the Gauss-Newton equation will not be a good approximation and the convergence may not be guaranteed. In such a case, it requires a better approximation of the Hessian given in (9).

C. SOFT-MAX

Soft-max regression is a generalized logistic regression that is used as a multi-class classification under the assumption that all the class labels are mutually exclusive.

$$p(y = j|h(x)^{(i)}) = \frac{e^{h(x)^{(i)}}}{\sum_{j=0}^{n} e^{h(x)^{(j)}}} \tag{13}$$

The net input $h(x)$ is:

$$h(x) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_m x_m = \sum_{i=0}^{m} \theta_i x_i = \theta^T x$$

where θ is the weight vector, x is the vector of one training sample, and θ_0 is the bias unit.

D. FEEDFORWARD NETWORKS

A deep feedforward network often called ANN, or MLP is the quintessential deep learning model. The goal of a feedforward network is to approximate some functions of F_1. For example, for a classifier, $y = F_1(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = F_1(x; \theta)$ and learns the value of the parameter θ that results in the best function approximation. These models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations used to define F_1, and finally to the output y.

Feedforward networks are of extreme importance to machine learning practitioners. They form the basis of many important commercial applications. For example, the convolutional networks used for object recognition from photos are a specialized kind of feedforward network. Feedforward networks are a conceptual stepping stone on the path to recurrent networks, which power many natural language applications. Feedforward NN are called networks because they are typically represented by comprising many different functions together [29].

IV. MATERIAL AND METHOD

In this study, a combined model of two super classifiers is presented: CNN and XGBoost. First, the CNN classifier is briefly introduced in section A and the XGBoost classifier in section B. The hybrid CNN-XGBoost trainable feature extractor model is presented in section C. Finally, the parameter selection and data preparation are discussed in section D.

A. CNN CLASSIFIER

Due to the availability of better computational hardware and massive training data in recent years, CNN has been used to achieve state-of-the-art performance in character recognition. CNN model can be used in three different ways: (i) training the CNN from scratch; (ii) transfer learning strategy to leverage features from a pre-trained model on a larger dataset; and (iii) keeping the transfer learning strategy and fine-tune the weights of CNN architecture [30]. A CNN is a multi-layer ANN with a deep supervised learning architecture that can be defined as the composition of mainly four layers.

1) CONVOLUTIONAL LAYER

A convolutional layer is the core building block of a convolutional network that does most of the heavy computational lifting. The input image is passed through a stack of convolution layers which is the main component of a CNN whose job is to detect important parts of the input image pixels. It is at this layer that all automatic feature extractions are performed using a window (patches) that moves over the image [31]. Many researchers use filters with a small 3×3 or 5×5 receptive field (which is the smallest size to capture the notion of left/right, up/down, and center) throughout the CNN or sometimes can be changed in between the architecture. In this paper, a 3×3 filter is considered and slide over the complete image. As shown in Fig. 1, a dot product between the filter and a small 3×3 region of the input image plus bias ($\theta^T x + b$) results in a scalar value. The output after the complete image is convolved is known as activation (feature) map. The spatial size of the output volume is computed as a function of the input volume size (W), the receptive field size of the Convolutional Layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border as below.

$$\frac{W - F + 2P}{S} + 1$$

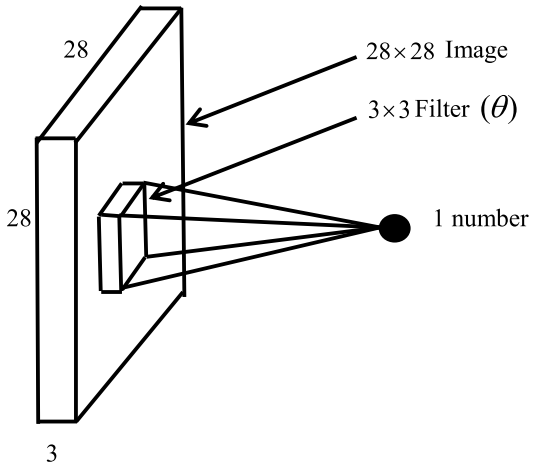


FIGURE 1. Convolving an image of 28 × 28 with a filter of 3 × 3.

A convolution layer can be mathematically described as follows:

$$C_p^l(i,j) = \sigma\Big(\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty} x(i-u, j-v) \times \theta_{p,q}^l(u,v) + b_q^l\Big) \quad (14)$$

where C is a convolution layer, l indicates the layer, p, q denote the map indices of current and next layers, respectively, and i, j are a row and column indices, respectively of the feature map. σ is the activation method, x is an image or activation map, θ is a kernel, and b is the bias [32].

2) POOLING LAYER

It is common to periodically insert a pooling layer inbetween successive convolutional layers in a convolutional network architecture. Its function is to progressively subsample the spatial size of the input image to reduce the number of parameters and computation in the network; and hence, control overfitting. The pooling layer operates independently on

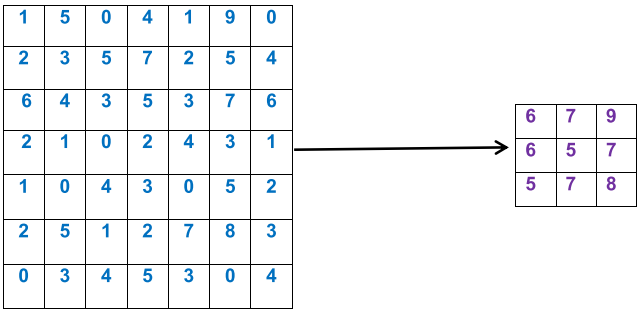


FIGURE 2. Max pool with 3 × 3 filter and 2 strides.

every depth slice of the input and resizes it spatially using the max operation. The down-sampling of the input size is also done the same way as a filter kernel without using zero-padding as below [33].

$\frac{W-F}{S} + 1$, where the same is true for the height. A pooling layer can be mathematically described as follows:

$$S_p^l(i,j) = \max\Big(\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty} C_p^l(i-u, j-v)\Big) \qquad (15)$$

The max value is chosen as in Fig. 2.

3) FULLY CONNECTED LAYER

As seen in regular ANN, neurons in a fully connected layer have full connections to all activations in the previous layer. A fully connected layer is a trainable classifier where their activations can be computed with a matrix multiplication followed by a bias offset. The only difference between the fully connected layer and convolution layer is that the neurons in the convolution layer are connected only to a local region in the input and that many of the neurons in a convolution volume share parameter. However, the neurons in both layers still compute dot products, so their functional form is identical [23]. The final output of the convolution layer will be an input to the fully connected layer for classification purposes.

4) RECTIFIED LINEAR UNITS

Although a number of activation methods exist, the most popular ones include: rectified linear units (ReLU), sigmoid (logistic), and hyperbolic tangent (Tanh). In a given network, the differentiability of an activation function is very important in order to perform a backpropagation mechanism. Accordingly, the weights can be optimized using any optimization techniques to reduce errors. Therefore, this paper uses a ReLU activation function. Mathematically, it is represented in the form of $h(x) = \max(0, x)$ where all negative elements of the input values (x) are clamped to zero. The rectifier activation function is used instead of a linear activation function to add non-linearity to the network. Otherwise, the network would only ever be able to compute a linear function.

Fig. 3 describes the general method followed in getting the best accuracy in comparing a fully connected layer and XGBoost as classifiers of the CNN. From this perspective as shown in Fig. 4, an eight-layer (only accounting for the convolutional layer and the fully connected layer) network consisting of six convolutional layers and two fully connected layers is designed. Every two convolutional and ReLU layers are followed by a max-pooling layer and a dropout layer. The last max-pooling layer is followed by a fully-connected layer that contains 512 neurons that are used to perform the final classification.

The input layer is a matrix of size $S_1 \times S_1$ RGB images. Feature map layers (L_1, L_2, L_3, L_4, L_5, and L_6) are used to compute the features, and every two feature maps used different resolutions. Each neuron on a feature map connects 9 inputs with its previous layers, and they are defined 3 × 3 convolutional filtering kernel known as the receptive field. All the neurons in one feature map share the same kernel and connect weights (known as the sharing weights). With a kernel size of 3 and a subsampling ratio of 2, each feature map layer reduces the feature size from the previous feature size S. The trainable classifier is the fully connected MLP, with a hidden layer (L_7) and an output layer (L_8). In this

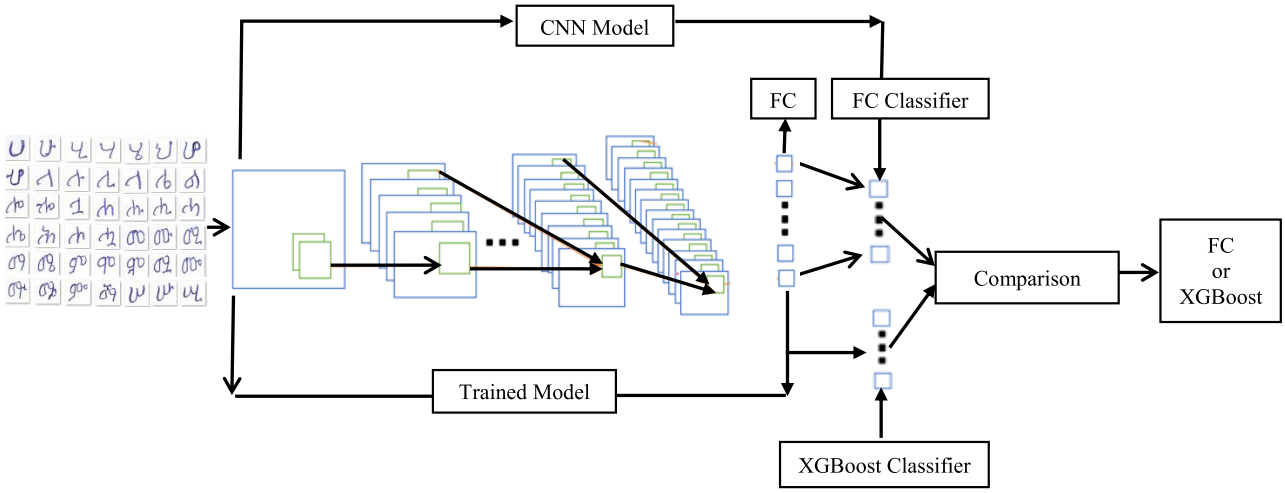


FIGURE 3. General diagram for FC and XGBoost classifiers. The above line with the CNN model shows the first model that has trained with the FC layer classifier, and the bellow line with the trained model uses the XGBoost classifier for classification. Both classifiers are then compared for classification.

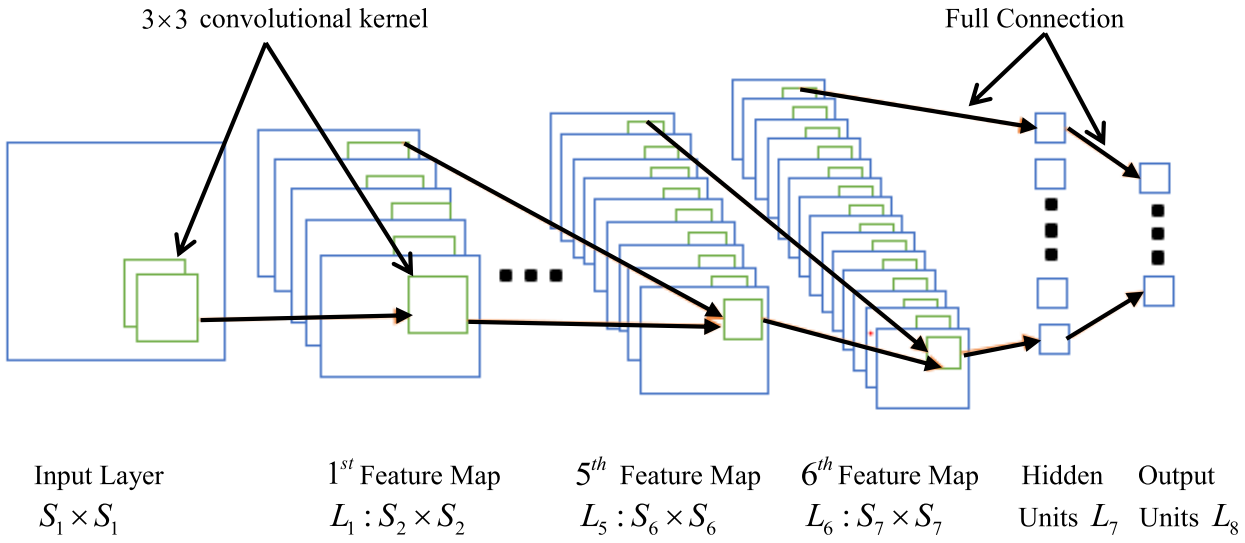


FIGURE 4. The original CNN model that has trained and is used as an input to the XGBoost classifier.

experiment, the last trainable classifier is modified by the XGBoost classifier and is described in detail in section B. The overall CNN architecture can be represented as

$$\begin{aligned} Input \rightarrow & \left[([Conv \rightarrow \mathrm{Re}LU] \times N) \rightarrow Pool^*\right] \times M \\ & \rightarrow [FC \rightarrow \mathrm{Re}LU] \times K \rightarrow FC, \end{aligned}$$

where N, M, and K are the number of times that can be repeated and $Pool^*$ represents the pooling layer that can be omitted (optional). Hence our CNN model is represented as follows.

$$\begin{aligned} Input \rightarrow & [([32Conv \rightarrow \mathrm{Re}LU] \times 2) \rightarrow MP \rightarrow Dropout] \\ & \rightarrow [([64Conv \rightarrow \mathrm{Re}LU] \times 2) \rightarrow MP \rightarrow Dropout] \\ & \rightarrow [([128Conv \rightarrow \mathrm{Re}LU] \times 2) \rightarrow MP \rightarrow Dropout] \\ & \rightarrow [512FC \rightarrow \mathrm{Re}LU] \rightarrow 502FC \rightarrow Output \end{aligned}$$

The CNN architecture is trained using FC as a classifier for 100 epochs. The learning rate and the decay factor are 0.001 and 0.9, respectively.

B. XGBoost CLASSIFIER

The idea of boosting came into existence when it was found, that a weak learner could be made a better learner using modification on assigning weights to the learners. The weak learner is a model whose performance is slightly better than random chance. Based on this concept, many improvements have been introduced so that boosting can have better performance. AdaBoost, ARCing, Gradient Boosting, and XGBoost algorithms are some of the modified boosting algorithms [34]. According to the concept of gradient boosting, it involves basically three steps: 1) a proper differentiable loss function is identified for a given problem. One benefit of the gradient boosting model is that for different loss functions, new algorithms are not required to be derived; it is enough that a suitable loss function is chosen and then incorporated with the gradient boosting framework; 2) a weak learner is designed to make the predictions. In gradient boosting, a decision tree is chosen as a weak learner; and 3) an additive model is created to add up the predictions of the weak learners so as to reduce the loss function. This process of adding the trees happens once at a time. The output produced in the new tree is then added to the output of the pre-existing sequence of trees in order to improve the final output of the model. This process stops once a desired value for the loss function is reached.

XGBoost also has gradient boosting at its core. Nevertheless, the difference between XGBoost and simple gradient boosting algorithm is that unlike gradient boosting, the process of combining weak learners does not happen one after the other. Therefore, it takes a multi-threaded approach where the CPU core of the machine is fully analyzed leading to greater speed and performance. Apart from that, there is a sparse aware implementation which also comprises automatic handling of missing data values, then block structure to support the parallelization of tree construction, and the process of continuous training so that one can further boost an already fitted model on new data. The XGBoost is a tree ensemble approach where multiple classifications and regression trees are combined [35]. For a given data set with n examples and m features $D = \{(x_i, y_i)\}(|D| = x_i \in \mathbb{R}^{n \times m}, y_i \in R^n)$, the mathematical representation of the tree ensemble model is represented in:

$$\hat{y}_i = \sum_{k=1}^{k} h_k(x_i), h_k \in R \tag{16}$$

where k is the number of trees, h is a function in the functional space R, and R is the set of all possible classification and regression trees. And the objective function is represented as:

$$F_1(\theta) = \sum_{i}^{n} h(y_i, \hat{y}) + \sum_{k=1}^{k} \Omega(h_k) \tag{17}$$

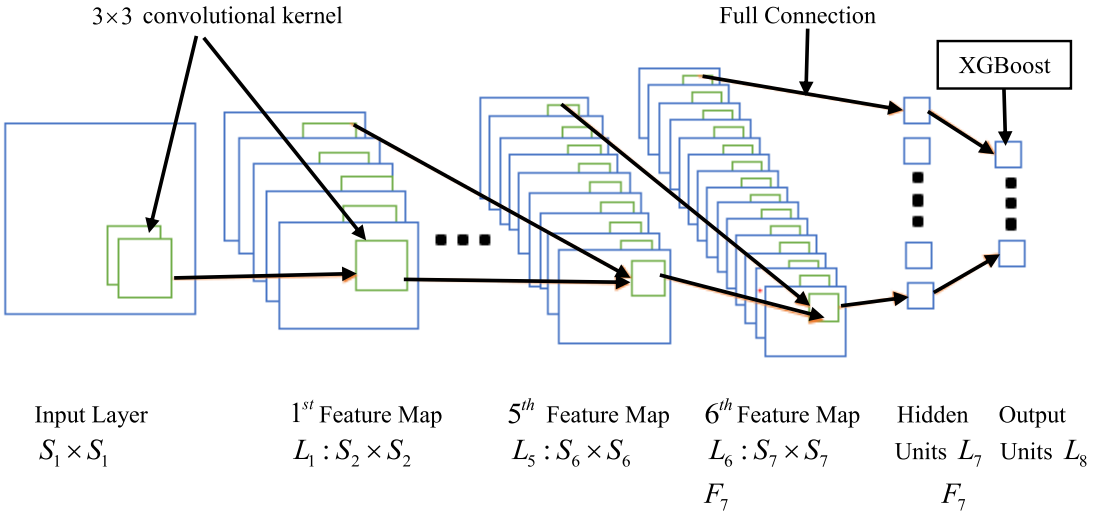


FIGURE 5. A new hybrid CNN-XGBoost model. The trained model up to the hidden unit F_7 is taken from Fig. 4. And F_7 is used as an input vector to the XGBoost classifier for final classification.

where $h(y_i, \hat{y}_i)$ is the training loss function, and $\Omega(h_k)$ is the regularization function, the goal of XGBoost is to minimize $F_1(\theta)$. Here h is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i. The second term penalizes the complexity of the model.

In XGBoost the regularization complexity can be defined as:

$$\Omega(h) = \gamma L + \frac{1}{2}\lambda \sum_{j=i}^{T} \omega_j^2 \tag{18}$$

where γ is the gamma parameter, L is the number of leaves, λ represents L_2 regularization term on weights in the model and, ω is the vector score on leaves,

The additional regularization term helps to smooth the final learned weights to avoid overfitting. Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.

Finally, the extracted feature outputs received from CNN's are fed to the XGBoost model for classification. Since XGBoost creates trees based on the number of labels, the time that will be spent on training will depend on the number of classes the dataset has.

C. HYBRID CNN-XGBoost MODEL

Feature extraction is the most important process in an automatic image classification system. The quality of extracted features can directly influence the recognition performance of the algorithm which was time-consuming in traditional image classification tasks. CNN is an efficient deep learning model with a hierarchical structure to extract and learn high-quality features at each layer. Since the model can reduce the complexity of network structure and the number of parameters through its properties like local receptive fields, weight sharing, and pooling operation, it has been widely used in image classification problems and achieved excellent results. Traditional classifiers connected to CNN do not fully understand the extracted features. Meanwhile, XGBoost is an integrated learning algorithm based on gradient boosting that can efficiently achieve higher classification accuracy; thus will overcome the limitations identified [36].

The architecture of our hybrid CNN–XGBoost model was designed by swapping the last fully connected layer of the CNN model with an XGBoost classifier. For output units of the last layer in the CNN network, they are the estimated probabilities for the input sample. Each output probability is considered by an activation function. The input of the activation function is the linear combination of the outputs from the previously hidden layer with trainable weights, plus a bias term. Looking at the output values of the hidden layers are worthless, but only makes sense to the CNN network itself; however, these values can be treated as features for any other classifiers. Fig. 5 shows the structure of the new hybrid CNN–XGBoost model. First, the normalized and uncentered input images are given to the input layer, and the original CNN model with the fully connected output layer is trained until the training process converges or the training error gets fixed. After the CNN model is trained, the last fully connected layer is removed. Second, the trained model from the first dense layer is taken as an input to the XGBoost classifier. The name of the dense layer is identified from the model summary. A new model is created to get the trained layers from that name *(new_model = model.get_layer ('dense_n').output)*, where 'dense_n' refers to the name of the dense layer. Using this trained model training, validation, and testing datasets are predicted. Both predicted values of training and validation datasets are given as an input feature vector to fit the XGBoost model. The XGBoost model has trained for 100 iterations. 70 early stopping rounds are fixed to avoid overfitting. The predicted testing dataset is also used for classification reports.

D. PARAMETER SELECTION AND DATA PREPARATION

To evaluate the feasibility of the hybrid model, experiments are conducted in the HECR dataset that is prepared for the first time. The dataset is prepared from 446 scripts (Fidel in Ethiopia), 20 numerical representations, 9 punctuations, 8 tonal symbols, 3 combining symbols, and 6 special characters, which contain 492 Ethiopian scripts. 10 Arabic numbers are also included in the dataset. The total number of scripts in the dataset becomes 502. All the scripts are distributed to different people who are of different backgrounds, ages, and sex. The dataset is prepared originally from 109 handwritten samples as it is depicted in Fig. 6. Table 1 shows some scripts most widely used in Ethiopia. As can be seen from Table 2 the detailed distribution of the dataset and its rotational style is presented. The images in the HECR dataset are RGB main color spaces that have been size normalized to 28×28 pixels.

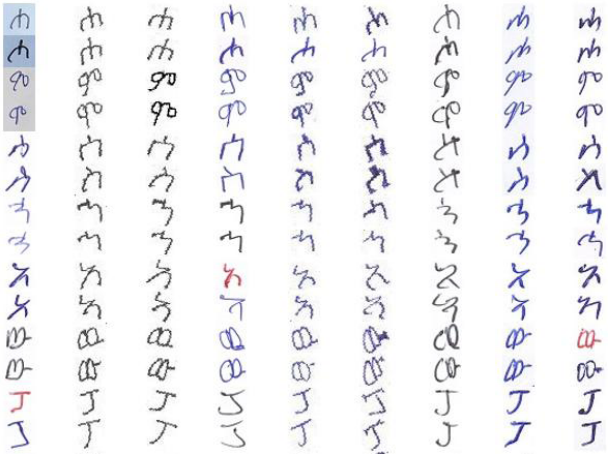


FIGURE 6. Sample images in the HECR dataset. Each row represents one type of script, whereas the column represents the number of participants.

TABLE 2. Dataset distribution and rotational angles.

Number of datasets	Type of Dataset
60560	Training set
15141	Validation set
10142	Testing set
85843	Total Dataset
29616	Rotated dataset
1506	Rotated + some noise
-20, -15, -10, -5, 5, 10, 15, 20	Rotational angle

Previous studies [37], [38] have verified that better generalization can be attained with an extended training dataset by using augmentation and distortion techniques. In this experiment, the transformation of the dataset is applied by implementing rotation, shifting, zooming, and adding noise in the CNN training phase.

The basic parameters used in this experiment are listed in Table 3. The parameters are selected based on the type of classification used. Since the research uses multi-class classification, the evaluation matrix and booster must be mlogloss and gbtree, respectively. The remaining parameters defined in the table are also very important, but for the rest of the parameters, default values of the library (XGBoost) are used.

TABLE 3. Parameters used in xgboost training.

XGBoost Parameters	Value
iteration	100
early_stopping	70
eta	0.3
eval_metrix	mlogloss
Booster	gbtree
gamma	0
max_depth	6

To compare human and machine script classification, a survey is conducted among 15 Ethiopian students studying at Xiamen University, China. This survey includes 16 selected scripts. Some of these scripts produced an error in the hybrid model. The remaining scripts are also selected based on the similarity of their structures.

V. RESULTS AND DISCUSSIONS

The experiments are organized in the following manner: Section A analyzes the results obtained using CNN and hybrid CNN–XGBoost model. Section B presents the reliability performance of the model versus human classification.

A. EXPERIMENTS ON THE HYBRID CNN-XGBoost MODEL

Fig. 7 depicts the output training accuracy from the CNN model that has used a fully connected layer as an output layer. Fig. 8 describes the error rate converged to a fixed value of 0.2188 in training. With this setup, the CNN learning classifier produced an error rate of 0.4630 on the testing dataset. Then, the new hybrid CNN–XGBoost model is built and trained.

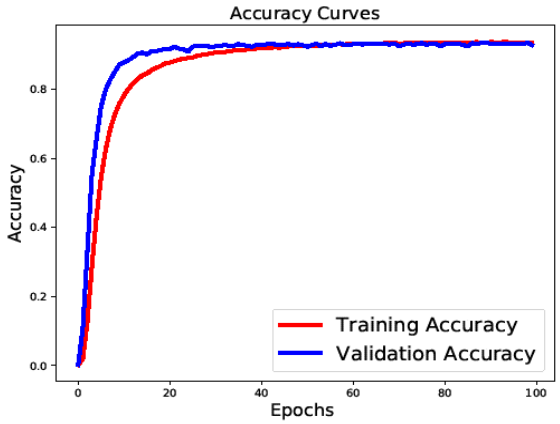


FIGURE 7. The training accuracy of the CNN model in the HECR dataset.

The last fully connected layer of CNN is replaced by an XGBoost classifier to predict labels of the input patterns. The trained output of CNN from the layer F_7 is used as a new feature vector to represent each input pattern and is

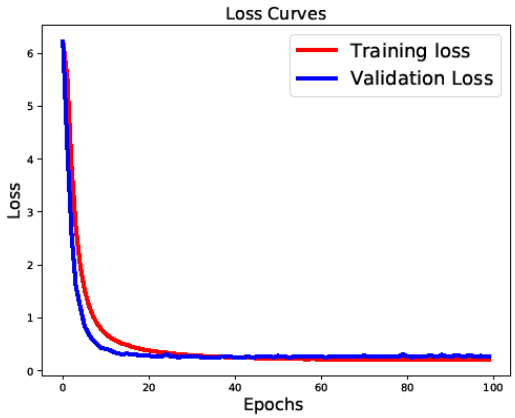


FIGURE 8. The training error of the CNN model in the HECR dataset classification.

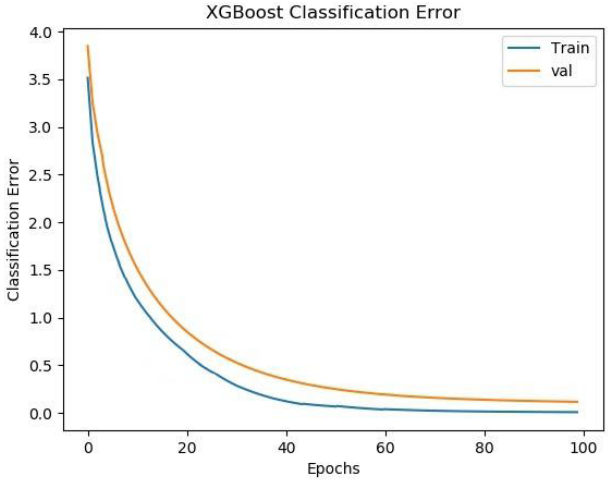


FIGURE 9. Classification error while XGBoost is used as a classifier in a trained CNN model.

TABLE 4. CNN and CNN-XGBoost classification error rate results in HECR dataset.

Error rate (%)	CNN	CNN-XGBoost
Training	0.2188	0.1334
Testing	0.4010	0.1612

fed to the XGBoost for training and testing. Fig. 9 shows the classification error during training. Table 4 lists all the training and testing error rates that occurred during the CNN and the hybrid model experiments.

To evaluate the performance of the new model, accuracy is not enough. Performance parameters like confusion matrix and classification report are also visualized. Table 5 shows the detail output of the classification report. All output parameters (precision, recall, and f_1-score) values are close to the classification accuracy. The output under support is the number of values whether they are predicted positive or negative in each category of the actual values. Comparisons with other results of different methods

TABLE 5. Classification report using XGBoost classifier.

	precision	recall	f1-score	support
Class 0	1.00	0.81	0.89	980
Class 50	0.95	0.90	0.93	1135
Class 100	0.83	0.95	0.88	1032
Class 150	0.91	1.00	0.95	1010
Class 200	0.86	0.95	0.90	958
Class 250	0.95	1.00	0.98	982
Class 300	0.58	0.70	0.64	892
Class 350	0.86	0.90	0.88	1028
Class 400	0.70	0.80	0.76	904
Class 450	1.00	1.00	1.00	1032
Class 500	0.99	0.99	0.99	1009
Class 501	0.99	1.00	0.99	1010
micro avg	0.95	0.96	0.94	10142
macro avg	0.96	0.98	0.97	10142
weighted avg	0.92	0.97	0.98	10142

TABLE 6. Comparison of accuracy on different datasets.

Reference	Method	Accuracy %
Niu [1]	CNN-SVM	99.81
Zhong *et al.* [29]	GoogleNet	99.83
Katiyar *et al.* [9]	SVM	97.16
Ramraj *et al.* [4]	XGBoost	99.56
James *et al.* [28]	CNN-XGBoost	97.18
Maitra [3]	CNN	99.10
Younis [12]	CNN	97.6
Proposed	CNN-XGBoost	**99.84**

published on the MNIST dataset are reported by Niu and Suen [1]. They chose the best recognition results generated by different learning algorithms with distortions applied to the training dataset. 0.19% was reported as the lowest error rate. However, a significant achievement is made by the proposed hybrid method with the lowest error rate of 0.16%, which boosted the performance by 15.789% compared with the best recognition result. Ramraj *et al.* [34] used the XGBoost classification method on the banknote authentication dataset. This hybrid outperforms by 0.28% accuracy. This indicates that XGBoost achieves higher classification accuracy when the final extracted features by CNN are given as input for classification. James *et al.* [39] used a hybrid of CNN-XGBoost for English characters and numbers that results in an output of 97.18% accuracy. Zhong *et al.* [40] used GoogleNet for Chinese character classification. Our proposed model achieves 2.66%, 0.01% improvements in the accuracy, respectively. In general, comparisons with different models published on different datasets are listed in Table 6.

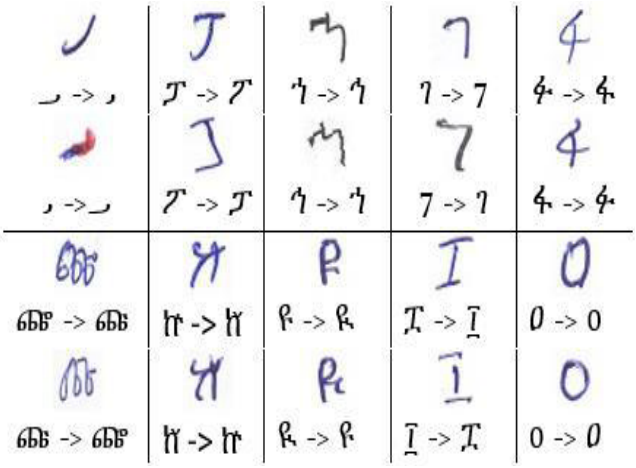


FIGURE 10. 20 script images that are mostly misclassified by the hybrid model. The lower labels of each image represent the corresponding Truth -> prediction.

Fig. 10 shows some misclassified samples of the dataset that are observed from the confusion matrix and classification report. After analyzing these errors, the cause of the misclassification is categorized into three types:

1) The most recurrent confusing pairs are listed in Fig. 10. They have similar shapes and structures in their nature. For example, the rightmost side of the figure is hard to recognize.
2) Some of the scripts have similar shapes and a structure due to people's writing habits. For example, when the images in Fig. 6 are closely examined, even humans cannot distinguish them without their labels. Even though every row represents different handwriting samples, the samples in every pair of the rows look very close to each other.
3) The degraded quality of the images, such as missing strokes, additional noises, rotations also have a great impact on the classification. The cases could be caused by people's poor handwriting, or caused by the scanning procedure, size normalization, and improper segmentation. For the third error category, it is extremely difficult for a machine to make a correct prediction with such ambiguous and low-quality inputs. Especially in this research paper, as many of the scripts have similarities, when a rotation is applied, some of the scripts have a high probability to be similar to another script from their orders.

TABLE 7. Confusion matrix on 16 scripts by 15 participants.

Truth		Participants																	%
Label	Shape	ሐ	ሖ	ም	ሞ	ሰ	ስ	ጎ	ጎ	አ	ኣ	ቁ	ቡ	ፓ	ፖ	ቸ	ኮ	Other	
ሐ		14																1	93.3
ሖ			14															1	93.3
ም				8	7														53.3
ሞ					15														100
ሰ						0	12											3	0
ስ							15												100
ጎ								10	5										66.7
ጎ								6	9										60
አ										10								5	66.7
ኣ											0							15	0
ቁ												3	12						20
ቡ													15						100
ፓ														14	1				93.3
ፖ														13	2				13.3
ቸ																8	7		53.3
ኮ																6	9		60

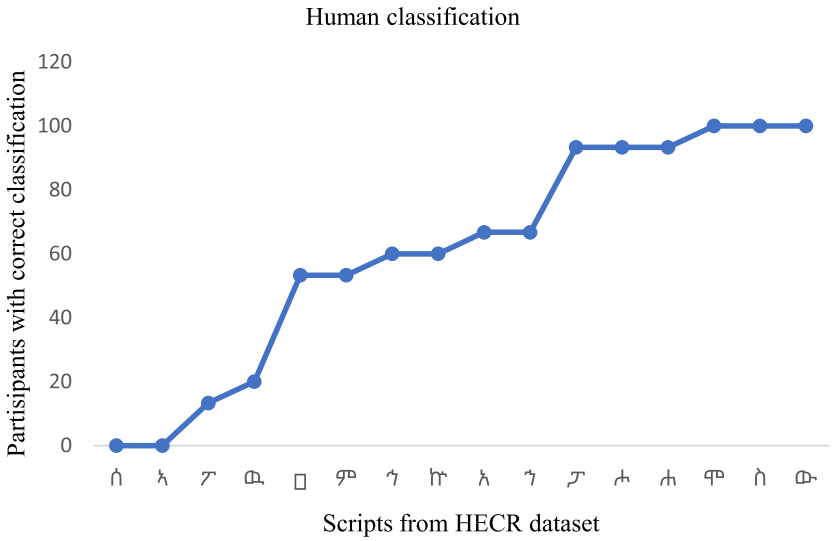


FIGURE 11. The percentage of correct classifications by the human on 16 testing images from the HECR dataset.

B. MACHINE RECOGNITION VERSUS HUMAN CLASSIFICATION

In this section, the differences between humans and machine in the recognition of handwritten characters are compared and discussed to evaluate the reliability of the proposed hybrid model. In order to conduct the survey, a classification report and confusion matrix is done using the new hybrid model on the testing dataset. Scripts that showed lower scores on the classification report (precision, recall, and f1_score), as well as a lower prediction score on the confusion matrix, are identified for comparison with the human classification survey. Table 7 summarizes the confusion matrix on 16 scripts by 15 participants. The ''Other'' column from the table represents a different script. Since all base scripts have 6 modified (order) scripts, thus, ''Other'' represents any of those orders or can be any other similar script. The correct labels of 16 sample script images are provided from the HECR dataset. From Fig. 11, 3/16 script images are properly classified by all the participants; and the remaining (13/16 script images) are recognized only by some members of the group. For those 3 scripts recognized correctly by all the participants, they are also correctly classified by the proposed hybrid model. 2 scripts are also completely misclassified by all participants. Especially row 10 is completely missed, even it is not classified as its pair (row 9) rather classified as ''Other''. In this case, the handwritten scripts that cause difficulty in being recognized by the majority of participants can also become difficult in being correctly classified by the machine. The details can be seen in Table 7 which shows the confusion matrix on the 16 scripts of the survey. From 16 scripts 75% of them are classified correctly above 50%. Even though row one, two, and nine are still classified as ''Others'', many of the participants have correctly classified them.

Upon examining the images closely, some of them are cursively written and there exists similarity between the images. Apart from the last two images, all images are from the same orders and it is clear that all the images in pairs have very close structures. This makes it hard for humans to identify what the scripts are without a ground truth, and the same is true for the machine. Another reason for misclassification by the hybrid model might be due to the lack of much training samples with similar structures as the number of images per script in the dataset are only 171. To solve this problem additional training samples are required. Rejection mechanisms can also be introduced.

VI. CONCLUSION

In this paper, a novel hybrid CNN–XGBoost model is proposed to solve the handwritten scripts recognition problem. In this integrated model, CNN works as a trainable automatic feature extractor from the raw images, whereas XGBoost performs the recognition and classification part. The competence and viability of the proposed model are evaluated in two aspects: the recognition accuracy and reliability performance. Experimental results in the HECR dataset show significant improvements attained by the proposed model.

Experimental results indicated that the proposed hybrid model is a promising classification method in the handwriting recognition field due to three properties: 1) The prominent features of the images can be automatically extracted by the hybrid model, whereas the success of most traditional classifiers relies largely on the retrieval of good hand-crafted features extractor which is a tedious and time-consuming task. 2) The hybrid model combines the qualities of CNN and XGBoost, as both algorithms are the most popular and successful classifiers in the handwritten character recognition field. 3) Even though the complexity of the hybrid model in the decision process is just increased a little bit when compared with the CNN classification model, the accuracy of the hybrid model is more promising, which is desirable when used in practical applications.

Research on the hybrid CNN–XGBoost learning model is still an active area. The performance of the hybrid model can be further improved through fine-tuning of its structure and its parameters. For example, improvements might be made based on the size of the input layers, the number of feature maps throughout the layers, the kernel functions used in the model, etc. Without being limited to the well-organized handwritten documents such as Chinese, English, Arabic, French, etc. it can also be further studied on recognition of all non-Latin handwritten characters, in different languages. As most valuable Ethiopian manuscripts have been written by non-Latin scripts, the manuscripts are diminishing at an alarming rate as they are gradually substituted by more recent ones and the rest are exposed to moth, fire, theft, and rupture. In order to protect and preserve the manuscripts and the history of Ethiopia from damage, document digitization must be done. The long-term plan for this research is going to develop a modern OCR application.

REFERENCES

[1] X. X. Niu and C. Y. Suen, "A novel hybrid CNN-SVM classifier for recognizing handwritten digits," *Pattern Recognit.*, vol. 45, no. 4, pp. 1318–1325, Sep. 2011.

[2] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, Jan. 2000.

[3] N. Aneja and S. Aneja, "Transfer learning using CNN for handwritten Devanagari character recognition," in *Proc. IEEE Int. Conf. Adv. Inf. Technol. (ICAIT)*, Sep. 2019, pp. 1–4.

[4] D. S. Maitra, U. Bhattacharya, and S. K. Parui, "CNN based common approach to handwritten character recognition of multiple scripts," presented at the Int. Conf. Document Anal. Recognit. (ICDAR), 2015.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf.*, vol. 3, Jun. 2016, pp. 1–13.

[6] H. S. Park and S. W. Lee, "Off-line recognition of large-set handwritten characters with multiple hidden Markov models," *Pattern Recognit.*, vol. 29, pp. 231–244, Feb. 1996.

[7] S. Purnamawati, D. Rachmawati, G. Lumanauw, R. F. Rahmat, and R. Taqyuddin, "Korean letter handwritten recognition using the deep convolutional neural network on the Android platform," presented at the 2nd Int. Conf. Comput. Appl. Inform., 2018.

[8] D. Keysers, T. Deselaers, H. A. Rowley, L.-L. Wang, and V. Carbune, "Multi-language online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1180–1194, Jun. 2017.

[9] J. Pradeep, E. Srinivasan, and S. Himavathi, "Diagonal feature extraction based handwritten character system using a neural network," *Int. J. Comput. Appl.*, vol. 8, pp. 17–22, Oct. 2010.

[10] G. Katiyar, A. Katiyar, and S. Mehfuz, "Off-line handwritten character recognition system using support vector machine," *Amer. J. Neural Netw. Appl.*, vol. 3, pp. 22–28, Dec. 2017.

[11] A. K. Tushar and A. Ashiquzzaman, "Handwritten Arabic numeral recognition using deep learning neural networks," presented at the IEEE Int. Conf. Imag., Vis. Pattern Recognit., 2017.

[12] K. S. Younis, "Arabic handwritten character recognition based on deep convolutional neural networks," *Jordanian J. Comput. Inf. Technol.*, vol. 3, pp. 186–200, Dec. 2017.

[13] P. M. Kamble and R. S. Hegadi, "Handwritten Marathi character recognition using R-HOG feature," presented at the Int. Conf. Adv. Comput. Technol. Appl. (ICACTA), 2015.

[14] J. Suganthi and R. Dineshkumar, "Sanskrit character recognition system using the neural network," *Indian J. Sci. Technol.*, vol. 8, no. 1, pp. 65–69, Jan. 2015.

[15] X. Chen, "Convolution neural networks for Chinese handwriting recognition," presented at the Stanford Univ., 2016.

[16] Y. Zhang, "Deep convolutional network for handwritten Chinese character recognition," presented at the Stanford Univ., 2016.

[17] F. D. Zewidie, "Developing character recognition for Ethiopic scripts," *Comput. Eng., Hogskolan Dalarna*, vol. 3, pp. 1–76, Jan. 2016.

[18] A. T. Birhanu and R. Sethuraman, "Artificial neural network approach to the development of OCR for real-life Amharic documents," *Int. J. Sci., Eng., Technol. Res.*, vol. 4, pp. 141–147, Jan. 2015.

[19] C. V. Jawahar and M. Meshesha, "Optical character recognition of Amharic documents," *Afr. J. Inf. Commun. Technol.*, vol. 3, no. 2, pp. 1–14, 2007.

[20] H. T. Weldegebriel, J. Chen, and D. Zhang, "Deep learning for Ethiopian Ge'ez script optical character recognition," in *Proc. 10th Int. Conf. Adv. Comput. Intell. (ICACI)*, Xiamen, China, Mar. 2018, pp. 540–545.

[21] G. Huang, Z. Liu, and L. Maaten, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jan. 2018, pp. 2261–2269.

[22] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," M.S. thesis, Informat., Math. Model., Tech. Univ. Denmark, Lyngby, Denmark, Mar. 2012.

[23] F. F. Li, "CS231n convolutional neural networks for visual recognition," in *Stanford Vision and Learning Lab*. Stanford, CA, USA: Stanford Univ. Press, 2019. [Online]. Available: http://cs231n.github.io/convolutional-networks/

[24] R. S. Olson, *Python Machine Learning*. Birmingham, U.K.: Packt, 2015. [Online]. Available: https://www.academia.edu/29079919/Python_Machine_Learning

[25] S. Guido and A. C. Müller, *Introduction to Machine Learning With Python*. Newton, MA, USA: O'Reilly Media, 2016. [Online]. Available: https://www.academia.edu/38935001/PDF_Introduction_To_Machine_Learning_With_Python_A_Guide_For_Data_Scientists_PDF_RR

[26] N. G. Andrew, "Lecture notes," unpublished.

[27] R. Anil, K. Manjusha, S. S. Kumar, and K. P. Soman, "Convolutional neural networks for the recognition of Malayalam characters," in *Proc. 3rd Int. Conf. Frontiers Intell. Comput.*, 2015, pp. 493–500.

[28] M. N. Kabir, "Numerical tools for some identification problems in industrial applications," Ph.D. dissertation, Dept. Comput. Math., Univ. Carolo, Brunswick, Germany, 2007. [Online]. Available: https://publikationsserver.tu-braunschweig.de/servlets/MCRFileNodeServlet/dbbs_derivate_00004464/dissertation.pdf

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, Oct. 2016. [Online]. Available: https://b-ok.org/book/3430720/13a073

[30] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten Arabic character recognition," *Cogn. Syst. Res.*, vol. 50, pp. 180–185, Aug. 2018.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional neural networks for large-scale image recognition," presented at the Int. Conf. Learn. Represent. (ICLR), Apr. 2015.

[32] Z. Zhang, "Derivation of backpropagation in a convolutional neural network," Oct. 2016.

[33] A. Lumini and L. Nanni, "Deep learning and transfer learning features for plankton classification," *Ecolog. Inform.*, vol. 51, pp. 33–43, Feb. 2019.

[34] S. Ramraj, N. Uzir, R. Sunil, and S. Banerjee, "Experimenting XGBoost algorithm for prediction and classification of different datasets," *Int. J. Control Theory Appl.*, vol. 9, no. 40, pp. 651–662, 2016.

[35] R. Song, S. Chen, B. Deng, and L. Li, *eXtreme Gradient Boosting for Identifying Individual Users Across Different Digital Devices*. Cham, Switzerland: Springer, 2016, pp. 43–54.

[36] X. Ren, H. Guo, S. Li, S. Wang, and J. Li, *A Novel Image Classification Method With CNN-XGBoost Model*. Cham, Switzerland: Springer, 2017, pp. 378–390.

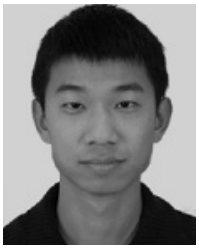
[37] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practice for convolutional neural networks applied to visual document analysis," in *Proc. Int. Conf. Document Anal. Recognit.*, Edinburgh, Scotland, 2003, pp. 958–962.

[38] X. Qu, W. Wang, K. Lu, and J. Zhou, "Data augmentation and directional feature maps extraction for in-air handwritten Chinese character recognition based on convolutional neural network," *Pattern Recognit. Lett.*, vol. 111, pp. 9–15, Aug. 2018.

[39] J. James, C. Lakshmi, U. Kiran, and Parthiban, "An efficient offline handwritten character recognition using CNN and XGBoost recognition using CNN and XGBoost," *Int. J. Innov. Technol. Exploring Eng.*, vol. 8, pp. 115–118, Apr. 2019.

[40] Z. Zhong, L. Jin, and Z. Xie, "High-performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps," presented at the Int. Conf. Document Anal. Recognit. (ICDAR), 2015, pp. 846–850.

HALEFOM TEKLE WELDEGEBRIEL received the B.Sc.Eng. degree in information technology and engineering from the Mekelle Institute of Technology and the M.Tech. degree in computer and information technology from the Defense University College of Engineering, Ethiopia, in 2009 and 2014, respectively. He is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, Xiamen University, Xiamen, China. His current research interests include optical character recognition, data mining, and big data using deep learning techniques.



HAN LIU received the B.Sc. degree in computing from the University of Portsmouth, in 2011, the M.Sc. degree in software engineering from the University of Southampton, in 2012, and the Ph.D. degree in machine learning from the University of Portsmouth, in 2015. He has been a Research Associate in computational intelligence with the School of Computing, University of Portsmouth. He is currently a Research Associate in data science and a member of the HateLaboratory and the Social Data Science Laboratory in the School of Computer Science and Informatics, Cardiff University. He has published two research monographs in Springer and over 50 articles in areas, such as data mining, machine learning, and intelligent systems. One of his articles was identified as a key scientific article contributing to scientific and engineering research excellence by the selection team at Advances in Engineering and the selection rate is less than 0.1%. He also has two articles selected, respectively, as finalists of Lotfi Zadeh Best Paper Award in the 16th and 17th International Conference on Machine Learning and Cybernetics (ICMLC 2017 & 2018). His research interests include data mining, machine learning, rule-based systems, intelligent systems, fuzzy systems, pattern recognition, big data, granular computing, and computational intelligence. He is a member of the Institution of Engineering and Technology (IET).

ANWAR UL HAQ received the bachelor's degree in computer science from the Department of Computer Science, University of Peshawar, in 2005, and the master's degree from the Department of Computer Science, The University of Manchester, U.K., in 2007. He is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, Xiamen University. His research interests include machine learning, decision support systems, and financial data mining.

EMMANUEL BUGINGO received the B.Sc. degree (Hons.) in information and communication technology from the University of Rwanda (Formal Umutara Polytechnic), Rwanda, in 2012, and the master's degree in computer science from Xiamen University, China, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Computer Science. His research interests include computer vision for image mining using deep learning, cloud computing, big data processing, and workflow scheduling.

DEFU ZHANG received the bachelor's and master's degrees in computational mathematics from Xiangtan University, in 1996 and 1999, respectively, and the Ph.D. degree in computer software and theory from the School of Computer Science, Huazhong University of Science and Technology. He was a Senior Researcher with the Shanghai Jinxin Financial Engineering Academe, from 2002 to 2003. He was a Postdoctoral Researcher with the Financial Data Mining Group, Longtop, from 2006 to 2008. From 2008 to 2016, he visited Hong Kong City University, University of Wisconsin Madison, and Macau University. Besides, he developed an Internet plus big data platform. He is currently a Professor with the Department of Computer Science, Xiamen University. He supervised the ACM/ICPC Team, Xiamen University. He has authored over 40 journal articles. His research interest includes computational intelligence, data mining, big data, cloud computing, online decision optimization, and food security. He was a recipient of three gold medals and eight silver medals, from 2004 to 2009, and he took part in the World Final Contest, in 2007.