IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

A Hybrid Deep Model for Recognizing Arabic Handwritten Characters

Naseem Alrobah[1] and Saleh Albahli[1,2]

[1]Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia
[2]Department of Computer Science, Kent State University, Kent, OH, USA.

Corresponding author: Saleh Albahli (e-mail: salbahli@qu.edu.com).

Handwriting recognition for computer systems has been in research for a long time, with different researchers having an extensive variety of methods at their disposal. The problem is that most of these experiments are done inEnglish, as it is the most spoken language in the world. But other languages such as Arabic, Mandarin, Spanish, French, and Russian also need research done on them since there are millions of people who speak them. In this work, recognizing and developing Arabic handwritten characters is proposed by cleaning the state-of-the-art Arabic dataset called Hijaa, developing Conventional Neural Network (CNN) with a hybrid model using Support Vector Machine (SVM) and eXtreme Gradient Boosting (XGBoost) classifiers. The CNN is used for feature extraction of the Arabic character images, which are then passed on to the Machine Learning classifiers. A recognition rate of up to 96.3% for 29classes is achieved, far surpassing the already state-of-the-art results of the Hijaa dataset.

INDEX TERMS Convolutional Neural Network, Machine Learning, Backpropagation, Arabic Character Recognition, Hijja Dataset, Optimizers.

I. INTRODUCTION

Handwritten language forms a timeless and inevitable aspect of human life. No matter how digital the world becomes, handwritten language will always remain. Even though the world is moving to a digital era, there are still some scenarios where the use of paper and pen cannot be avoided.

Character recognition technologies provide the users an automatic mechanism for recognizing the text on the image and converting the characters to their corresponding digital format. They are used in many verification applications, e.g., verifying official documents and bank cheques and helping visually impaired people read, e.g., reading paper currency or street signs. Character recognition system can be used to read both typed and handwritten scripts. The handwriting varies, especially in cursive script, where the writers' handwritten characters' size and style vary. Therefore, handwriting recognition is considered a more difficult task in computer vision [1].

Handwritten character recognition has been under study for a long time, with different researchers having many methods at their disposal. However, the problem is the majority of these experiments are done in Latin script, e.g., English, as it is the most popular language in the world [2]. Therefore, there are many available character recognition applications for English language. On the other hand, Arabic is one of the top-5 spoken languages globally, with 315 million native speakers [3], which means now, more than ever, computers need to also understand what these speakers are writing. However, due to the Arabic script's unique characteristics, e.g., cursive nature, presence of diacritics, and diagonal strokes, as well as the absence of publicly available standard datasets, developing an Arabic character recognition system is still challenging [4].

Recently, Arabic Handwritten Character Recognition (AHCR) became an active area in pattern recognition and computer vision. AHCR technologies improved significantly by using different Machine Learning (ML) algorithms, such as Support Vector Machines (SVM) and Artificial Neural Networks (ANN). However, the AHCR models that use Convolutional Neural Networks (CNNs) achieved the state-of-the-art results in different Arabic handwritten datasets[5] [1] [6]. CNNs automatically detect and extract the distinctive and representative features of the images, outperforming the classical ML algorithms that need to manually define the features (e.g., SVM [7] [8]). Therefore, CNNs achieve better results with large datasets

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

that have a large number of classes. At the classification stage, Niu and Suen [9] claim that SVM provides better results when it is replaced with MLP at the classification stage in deep learning. They explain the reason isthat MLP is based on the Empirical Risk Minimization, which attempts to minimize the errors in the training set. Therefore, the training process is stopped when the first separating hyperplane is found by the back-propagation algorithm. On the other hand, an SVM classifier aims to minimize the generalization errors by using the Structural Risk Minimization principle. The algorithm tries to find the hyperplane that maximizes the margin area between two classes of training samples. Therefore, the generalization ability of SVM is maximized to enhance the classification accuracy of the hybrid model after replacing the MLP output units in the CNN.

SVM is a binary classifier natively but can also be used for the case of multi-class classification. For multi-class classification, the same principal of binary classification is used, but SVM changes the problem into a multiple binary-classification problem. The main idea is to map the data into high-dimensional space and apply mutual linear separation between two classes.

To summarize, CNNs outperform conventional ML classifiers in feature extraction, while the latter outperform CNNs in classification. Therefore, different hybrid CNN architectures have been proposed [10] to compensate the limitations of CNN and conventional ML classifiers by integrating their features.

The hybrid CNN architectures can be defined as the CNN architectures that replace the softmax fully connected layer (FCL) of the CNNs with a classical ML classification algorithm [9]. Therefore, our main contribution in this work is to study the recognition of Arabic handwritten into different hybrid CNN architectures to show how the state-of-the-art methods can be enhanced based on hybrid models. Therefore, we study the effect of different aspects of hybrid CNN architecture on the AHCR performance. They include the base CNN architectures, the ML classification algorithms, and the optimizers and weight initializers that are used to train the models.

The rest of the paper is organized as follows: Section 2 reviews the AHCR state-of-the-art, and Section 3 presents the methodology Section 4 describe the experiment. Section 5 presents and discusses the experimental results, and Section 6 concludes the paper.

II. LITERATURE REVIEW

The state-of-the-art deep learning models in AHCR can be broadly categorized to standalone CNN models and hybrid CNN models. The following subsections outline them with tables to summarize them.

A. AHCR CNNs Models

El-Sawy et al. [11] claim that CNN approaches outperform other methods in feature extraction and classification tasks, and that they are suitable for use in Arabic character-recognition models. However, they requir a large dataset to achieve good results, while the available Arabic Handwritten Characters datasets are limited in the number of images. Therefore, they developed Arabic Handwritten Characters Dataset (AHCD) and designed a CNN model. The CNN approach provides an accuracy of 94.9%. Similar to the previous study, Altwaijry et al. in [1] introduced the Hijja dataset, a new Arabic handwritten characters dataset written by children aged from 7 to 12 years old. They also proposed a CNN-based model for recognizing Arabic handwritten characters. To evaluate their model, they compared its performance with the one introduced in El-Sawy's paper [11] on both AHCD and Hijja datasets. From the empirical results, Altwaijry's model outperformed the other model with an accuracy of 88% and 97% on Hijja and AHCD datasets, respectively. While the other model achieved an accuracy of 80% and 93.84% on Hijja and AHCD datasets, respectively. Balaha et al. in [12] introduced an Arabic handwritten character dataset (HMDB) and two CNN-based architectures, HMB1 and HMB2, where the former has a larger number of hidden layers and trainable parameters to investigate the impact of the architecture complexity in recognizing Arabic handwritten script. The two proposed architectures were trained and tested on three datasets: HMBD, CMATER, and AIA9k. Through 16 experiments, the effect of data augmentation, regularization, weight initializers and optimizers on the models' performance were studied. The best results were 90.7%, 97.3% and 98.4% for HMBD, CMATER, and AIA9k, respectively. They compared their architecture with the one proposed in [11] by training and testing it on HMBD—the accuracy of the model did not exceed 47.7%.

Boufenar et al. in [7] used Alexnet, a popular CNN architecture that consists of 5 convolutional layers and 3 max-pooling layers followed by 3 fully connected layers. They studied the effect of preprocessing in improving the model results. The Alexnet model was trained and evaluated in two datasets, OIHACDB-40 and AHCD, with three learning strategies: training the CNN model from scratch, using transfer-learning strategy and fine-tuning the CNN. The experimental results showed that the first strategy outperformed the others with 100% and 99.98% accuracies for OIHACDB-40 and AHCD, respectively.

VGG16 [13] is a very deep CNN architecture that was built to investigate how the network depth affects the accuracy in a large-scale image-recognition setting. It consists of 16 layers and is used in different Handwritten recognition applications. De Sousa [6] and Mudhsh [8] designed models for Arabic handwritten character recognition that were inspired by VGG-16. De Sousa in [6] proposed two deep CNN-based models, VGG-12 and REGU, for recognizing Arabic Handwritten characters and digits. The VGG-12 model is inspired by VGG-16 by removing the fifth convolutional block and adding a

dropout layer before the softmax FCL that is used as a classifier. While the REGU

Table 1. An overview of works related to CNN architectures in AHCR

Paper	Year	Dataset	Method	Optimization	Result in Accuracy
El-sawy et al. [11]	2017	AHCD	CNN	Mini-batch	94.9
Altwaijry et al. [4]	2017	AHCD	CNN	Dropout Weight Decay	97%
		Hijja [4]			88%
Balaha et al. [27]	2020	HMBD [15]	CNN	Dropout	90.7%
		AHCD [38]			97.3%
		AIA9k [14]			98.4%
Boufenar et al. [7]	2017	AHCD [38]	CNN (Alexn et)	Dropout Mini-batch	99.98%
		OIHAC DB-40 [5]			100%
De Sousa [6]	2018	AHCD [11]	CNN (based on VGG)	-Dropout. -Batch Normalization -Data Augmentation : zoom, translation.	98.42%.
Mudhsh et al. [8]	2017	HACDB	CNN (based on VGG)	-Dropout -Data Augmentation	97.32%
Alyahya et al. [15]	2020	AHCD [11]	CNN	-	98.30%

model was designed from scratch by adding different dropout and batch-normalization layers at both CNN and the Fully Connected Layers. The two models were trained once without data augmentation and once with data augmentation. Then, an ensemble model of the four models was created by averaging the predictions of each of the 4 models. The best test accuracy of the ensemble model was 99.47%. Mudhsh et al. in [8] proposed a model for Arabic handwritten digits and characters recognition. The model consists of thirteen convolutional layers, two max-pooling layers, followed by three Fully Connected Layers. To reduce model complexity and training time, only one-eighth of the filter in each layer of the original VGG-16 was used in the proposed model. Two different datasets were used to train and evaluate the model: ADBase was used for the digit recognition task, while HACDB was used for the character recognition task. To avoid the overfitting problem, they used dropout and data augmentation. The achieved accuracy of the model using the ADBase database was 99.66% and 97.32% for the HACDB database.

Alyahya et al. in [15] investigated the performance of ResNet-18 architecture in recognizing Arabic handwritten characters when FCL and Dropout are added to the original architecture. They designed 4 deep models: 2 models that used a Fully Connected Layer with/without dropout layer after all convolutional layers and 2 models that used 2 Fully Connected Layers with/without a dropout layer. They used AHCD dataset to train and evaluate the CNN based ResNet-18 model and the best test result was 98.30%, achieved by the original ResNet-18 in AHCR. Table 1 summarizes the CNN state-of-the-art models. The table illustrates the author of each article, the year of publication, the dataset used for training and evaluating the model, the optimization techniques used to improve the model and the testing result in terms of accuracy.

B. AHCR Hybrid CNNs Models

Elluch et al. [16] claimed that CNN achieves excellent results in term of feature extraction; however, they use Fully Connected Layers (FCLs) as the last layer to perform the classification task. They designed a CNN-SVM hybrid model and trained it with the HACDB dataset. The model achieved 6.59% Error Classification Rate (ECR), whereas the standard CNN model achieved an ECR of 14.71%. Also, Elluch et al. in [17] improved the previous model by adding the dropout technique. The ECR is reduced to be 5.83%.

Shams et al in [18] proposed a deep learning model composed of CNN with FCL, then the SVM were used to boost the results of the FCL. They focused on solving the problem of multi-stroke Arabic characters recognition, i.e. recognition of letters that have the same stroke but are different in the number and position of the dot(s), by using unsupervised K-mean clustering algorithm to group the characters into 13 clusters. They used AHCD [11] for training and evaluating the model. The model achieved 95.07% in term of accuracy outperforming Elsawy [11] model that achieved 94.90%.

The details of hybrid approaches for recognizing isolated Arabic Handwritten Characters are summarized in Table 2. The results in this table are presented in terms of Error Character Rate.

Table 2. An overview of works related to Hybrid CNN Architectures in AHCR

Paper	Year	Dataset	Hybrid Model Architecture	Results in ECR
Elluch et al. [16]	2016	HACDB	CNN+SVM	6.59%
Elluch et al. [17]	2016	HACDB	CNN+SVM	5.83 %.
Shams et al. [18]	2020	AHCD [11]	CNN + FCL + SVM	4.93%

ECR: Error Character Rate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

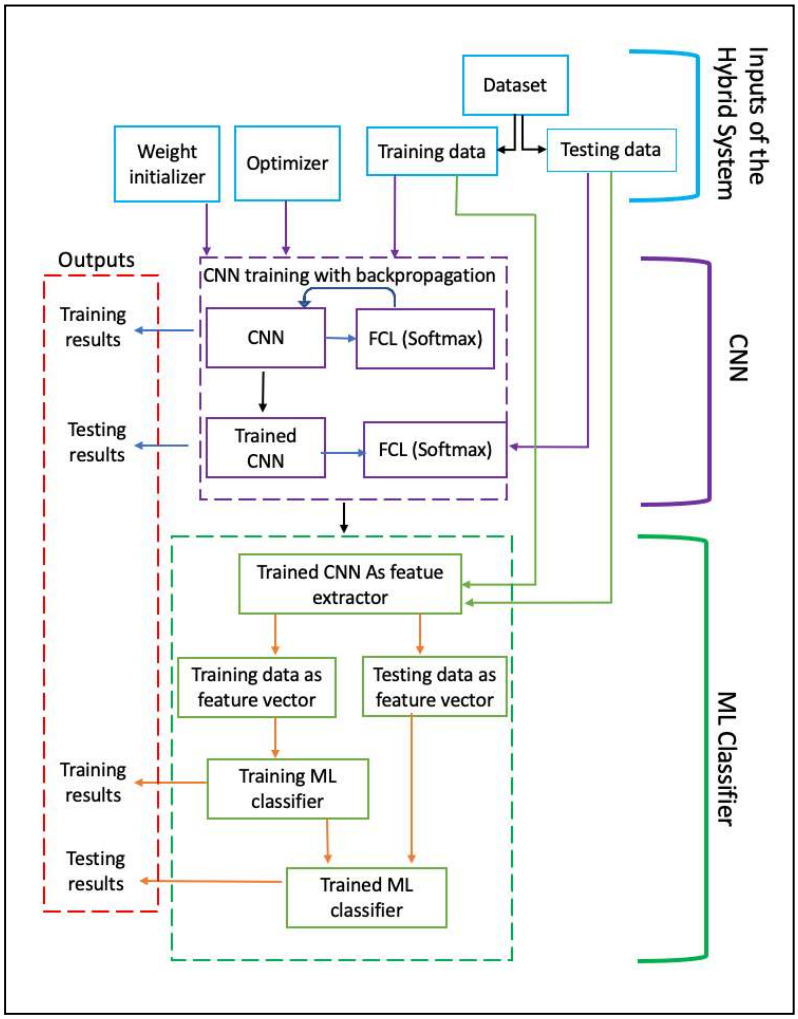

Figure 1. General Diagram for Hybrid CNN System

I. METHODOLOGY

comprises CNN architecture and ML classifiers. However, CNN trained with backpropagation and cannot be directly trained with the ML classifier. Therefore, the CNNs trained first with the softmax FCL then used as feature extractor for training and testing the ML classifier. A general framework of the hybrid CNN system is introduced in the first subsection. The second subsection elaborates the component of hybrid CNN architecture used in this study and the enhanced training procedure of hybrid models. The third subsection introduces the experiment design of evaluating the CNN hybrid model.

A. HYBRID CNN FRAMEWORK DESCRIPTION

Figure 1 describes how the proposed hybrid CNN system works. It is composed of four parts: the inputs of the system, a CNN for feature extraction, an ML classifier that takes the extracted features from CNN as input and uses them to perform classification, and the outputs of the system. The top section represents the inputs to the system, which includes the data, optimizer and weight initializer used as inputs to CNN. It can be seen that the dataset is split for training and testing. The middle section illustrates the CNN model and the lower represents the ML classifiers. After training the CNN, it is evaluated by the testing data and used in the next part as feature extractor.

In the ML classifiers section, the previously trained CNN receives the training and testing data and transforms them into feature vectors. Then the ML classifiers are trained and evaluated by these feature vectors. The outputs of the system are the training and testing results of the CNN model and CNN hybrid model.

B. Hybrid CNN Architecture

1) CNN ARCHITECTURES

As mentioned earlier, the hybrid CNN architecture

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

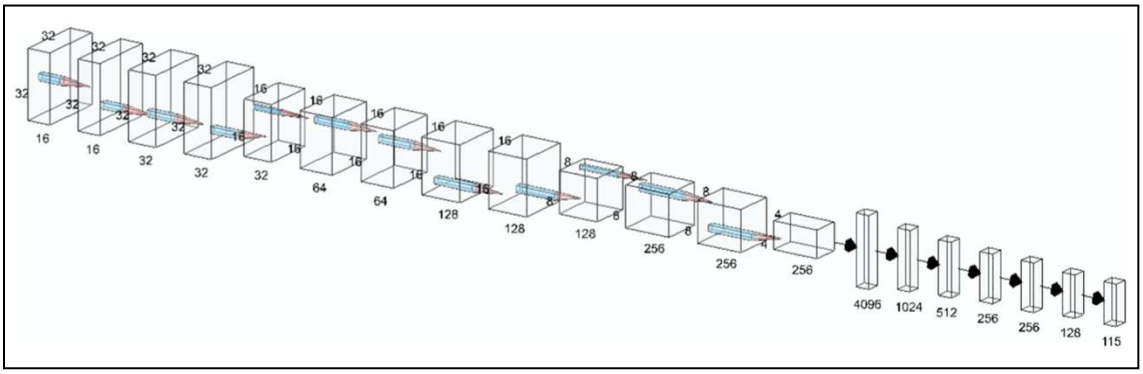

Figure 3. **HMB1 Architecture from [27]**

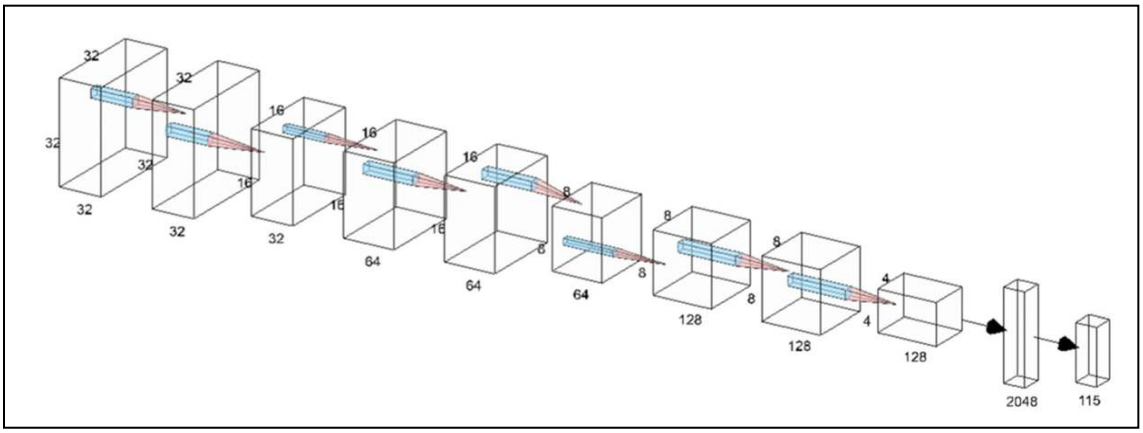

Figure 2. **HMB2 Architecture from [27]**

In this study, we consider 2 deep CNN architectures: HMB1, and HMB2[19], as they are CNN architectures of the top-performing architectures from related works evaluated on the AHCD dataset [20]. The CNN models, which consist of alternating layers of convolutional, activation, and pooling layers, are used for feature extraction. The following two subsections explain the details of CNNs layers.

a: HMB1

The input to the model is 32×32 images. All the convolutional layers used are kernel size of 3×3 while the max-pooling layers used pool size of 2×2. It includes 13 convolutional blocks where each block has a different arrangement of convolutional, batch normalization and pooling layers. These blocks are followed by 5 FCL with ReLU activation function. The original output layer of the model is an FCL with 115 neurons, and softmax is used as an activation function. To improve the model, both dropout with a ratio of 0.25 and Batch Normalization are applied to reduce the overfitting. The total number of parameters in HMB1 is 6,076,525, and 6,071,693 parameters out of them are trainable. Therefore, it is considered as a complex CNN architecture. Figure 2 and table 3 depict the design of the HMB1 architecture.

b: HMB2

Same as HMB1, the input to HMB2 is 32×32 images. As shown in figure 3, HMB2 includes 9 convolutional blocks. different arrangements of convolutional, max-pooling, batch normalization, and dropout layers used in the blocks. After the convolutional blocks, a flattened layer of 2048 neurons was used as a connection between the convolutional blocks and the FCL classifier. A total number of parameters is 181,373, and 179,869 out of them are trainable parameters. Therefore, it is considered less complex architecture than HMB1. Figure 3 and table 4 depict the design of the HMB2 architecture.

Table 3. **Architecture of HMB1**

Layer Type	Output Shape	Layer Type	Output Shape
Convolutional Layer	(32, 32, 16)	Batch Normalization	(8, 8, 256)
Convolutional Layer	(32, 32, 16)	Max-Pooling Layer	(4, 4, 256)
Batch Normalization	(32, 32, 16)	Flatten Layer	(4096)
Convolutional Layer	(32, 32, 32)	Dense Layer	(1024)
Convolutional Layer	(32, 32, 32)	Batch Normalization	(1024)
Batch Normalization	(32, 32, 32)	Dropout Layer	(1024)
Max-Pooling Layer	(16, 16, 32)	Dense Layer	(512)
Convolutional Layer	(16, 16, 64)	Batch Normalization	(512)
Convolutional Layer	(16, 16, 64)	Dropout Layer	(512)
Batch Normalization	(16, 16, 64)	Dense Layer	(256)
Convolutional Layer	(16, 16, 128)	Batch Normalization	(256)
Convolutional Layer	(16, 16, 32)	Dropout Layer	(256)
Batch Normalization	(16, 16, 64)	Dense Layer	(128)
Max-Pooling Layer	(16, 16, 64)	Batch Normalization	(128)
Convolutional Layer	(16, 16, 64)	Dropout Layer	(128)
Convolutional Layer	(16, 16, 128)	Dense Layer	(115)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

Table 4. Architecture of HMB2

Layer Type	Output Shape	Layer Type	Output Shape
Convolutional Layer	(32, 32, 32)	Max-Pooling Layer	(8, 8, 64)
Batch Normalization	(32, 32, 32)	Dropout Layer	(8, 8, 64)
Convolutional Layer	(32, 32, 32)	Convolutional Layer	(8, 8, 128)
Batch Normalization	(32, 32, 32)	Batch Normalization	(8, 8, 128)
Max-Pooling Layer	(16, 16, 32)	Convolutional Layer	(8, 8, 128)
Dropout Layer	(16, 16, 32)	Batch Normalization	(8, 8, 128)
Convolutional Layer	(16, 16, 64)	Max-Pooling Layer	(4, 4, 128)
Batch Normalization	(16, 16, 64)	Dropout Layer	(4, 4, 128)
Convolutional Layer	(16, 16, 64)	Flatten Layer	(2048)
Batch Normalization	(16, 16, 64)	Dense Layer	(115)

2) ML CLASSIFIERS

The extracted features from the trained CNN are used as inputs to 2 different classifiers: SVM and XGBoost to perform the classification tasks.

a: SUPPORT VECTOR MACHINE (SVM)

SVM is a machine-learning algorithm that can be used for both classification and regression. It has been used in many fields to achieve state-of-the-art results. SVM works by assuming that training examples are plotted in high-dimensional feature space. The role of SVM is to find hyperplanes that perfectly divide the dataset samples. Kenel, C and gamma are the parameters that are used in building the SVM models, and the performance highly depends on them. In this work, the output of the CNN network is a matrix that is flattened to be a vector of features. SVM is a well-known classifier for its effectiveness in classifying data with high dimensional features. Therefore, we chose SVM as a classifier in the hybrid models.

b: EXTREME GRADIENT BOOSTING (XGBOOST)

Many boosting classifiers are available as free open-source ML libraries, e.g., Adaptive Boosting (Adaboost) [21] and gradient boosting [22]. The basic idea behind boosting [23] is adding weak learners sequentially to the classification model to correct previously misclassified training samples and enhance the overall model performance. The weak learner is the model whose performance is slightly better than random guessing. The gradient boosting is a boosting algorithm that uses gradient descent to minimize the cost function. XGBoost is an improved and scalable implementation of a gradient boosting framework. Since it offers enhanced features such as faster speed, customization of the objective function and evaluation function, plus better overall performance [24], it was chosen for use in this study.

3) TRAINING PROCEDURE OF HYBRID CNN MODELS

The previous subsections present the details of the hybrid architecture components: CNN architectures and the ML classification algorithms. The basic idea behind the hybrid CNN models is replacing the original CNN output layer (softmax FCL) of the CNN with an ML classifier. This section illustrates the enhanced two-stage training procedure of the hybrid CNN models.

- First Stage (training the CNN with backpropagation): the convolutional layers in CNN learn their weights during the training by backpropagating the error from the output layer back through the FCLs to the first convolutional layer. Due to the nature of the learning process, the FCL should be used as an output layer during the training of CNN. The output of the CNN after this step is the features vector, which is the linear combination of the outputs from the previous hidden FCL with trainable weights, plus a bias term. It seems meaningless to humans, but it makes sense to be used as features for any classifier [9].
- Second Stage (training the ML classifier): the output FCL in the trained CNN is replaced with the ML classifier, namely SVM or XGBoost. The classifier is trained with the feature vector produced by the trained CNN. They are trained by the same training data used to train the CNN model.

The stages of the enhanced training procedure are illustrated in figure 4 and figure 5.

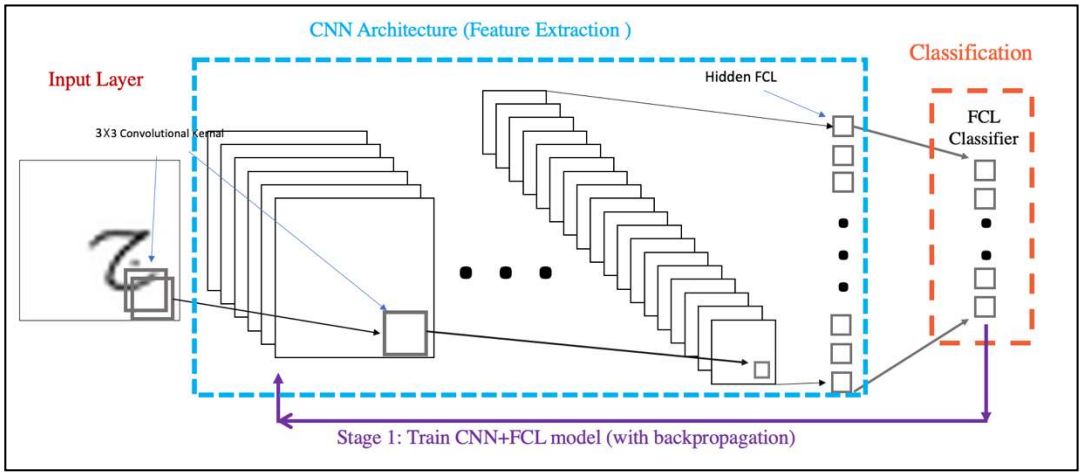

Figure 4. First Stage of Training Hybrid Models

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

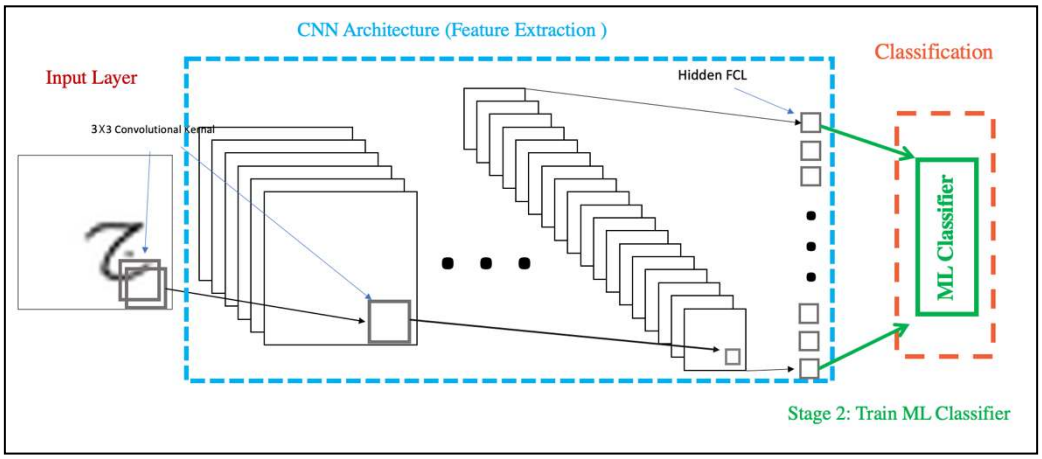

Figure 5. **Second Stage of Training Hybrid Models**

C. Experiment Design

In this study, extensive experiments were designed to study the effect of different CNN architectures, ML classifiers, and optimizers and weight initializers on the performance of the hybrid models. 30 different combinations of 5 optimizers (Adam [25], AdaDelta [26], AdaMax [27], AdaGrad [28] and stochastic gradient descent (SGD) [29]) and 6 weights initializers (Glorot uniform and Glorot normal initializers [30], He uniform and He normal initializers [31] and LeCun uniform and LeCun normal initializer [32]) were used in training the hybrid models. We used accuracy as a performance metric since the used dataset is a balanced dataset, i.e., the number of samples in each class is equal. The pseudocode below illustrates the experiment procedures. It should be mentioned that the enhanced 2-stage procedure is part of the experiment.

For optimizer in optimizers lists **do**
  For initializer in weight initializers list **do**
    Train the CNN with the selected optimizer and initializer using training data
    Evaluate the CNN with testing data
    For classifier in ML classifiers list **do**
      Replace the last FCL with classifier
      Train the classifier using training set
      Evaluate the classifier with testing set
    End_For
  End_For
End_For

III. EXPERIMENTS

This section introduced the experiments that are conducted on AHCR hybrid CNN architectures. It includes the dataset and the training parameters.

A. DATASET

The Hijja dataset is a free and publicly available dataset collected by Altwaijry et al [1]. It includes images of separated Arabic handwritten letters collected from children between the ages of 7 and 12. The dataset has 108 classes, which represent the Arabic letter in 4 shapes: beginning, middle, and end of the word and letter in isolated shape. The size of the image is 32*32 pixels. The total number of images in the dataset is 47,434. The hybrid CNN architectures are trained and evaluated using Hijja dataset. We used a total of 12,776 images that belong to the letters in isolated shape, with 29 classes corresponding to the Arabic letters in addition to "Hamza". The dataset is randomly split into 80% for training and validation (the 80% set is split internally into 80% for training and 20% for validation) and 20% for testing. Thus 8,174 images are used for training, 2,059 images for validation and 2,543 images for testing. Table 5 shows an image sample with the corresponding Arabic typed character for each class in Hijja dataset. The participants of Hijja are children and the resolution of the images is very low. Therefore, it has many distorted or unclear characters. Figure 6 shows distorted samples of Arabic letters. The character Kaf "ك" and the character Faa "ف" can be misclassified with Noon "ن", the character Meem "م" can be misclassified with Ha "ح", and the characters Haa "هـ" and Qaf "ق" are distorted and unreadable. Hijja is a new dataset that was published in 2020 and there are not many experiments on it. However, as far as we know, the best result has been achieved with the Hijja dataset was 88% in terms of accuracy. Therefore, in this experiment, the Hijja dataset has been chosen to fairly evaluate the CNN hybrid models.

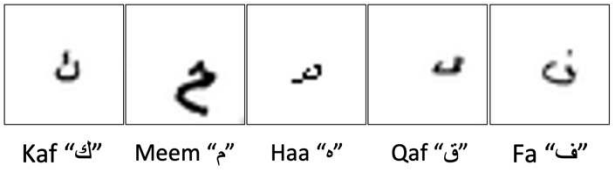

Figure 6. **Distorted and Unclear Samples from Hijja Dataset**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

Table 5. Samples of Hijja Classes

									
أ	ب	ت	ث	ج	ح	خ	د	ذ	ر
									
ز	س	ش	ص	ض	ط	ظ	ع	غ	ف
									
ق	ك	ل	م	ن	هـ	و	ي	ء	

Google Colaboratory (Colab)) [33] was chosen to implement the Hybrid CNN architecture and conduct the experiments. For this research, we used different open-source Python libraries and frameworks to deal with the images, implement the DL and ML models. They are:

- Tensorflow [34] for implementation and evaluation of the CNN models.
- Scikit-learn for dealing with multi-dimensional arrays and implementing ML models.
- OpenCV was used widely through the study for reading, resizing, and saving images.
- CSV library provides methods to deal with the CSV (Comma Separated Values) format which is used to save and analyze the experiment results.
- Glob is used in the experiments to find all the file names that follow a specific pattern.

B. TRAINING PARAMETERS

In this study, we trained and evaluated the CNN hybrid models separately, but using the same parameters and metrics. Relu [35] is used as an activation function for all convolutional and hidden FC layers, while softmax [36] is used for the output layer in CNN-FCL architecture. Max pooling is chosen to be used in the pooling layer as they are preferable in handwritten recognition tasks. We trained the model on different epochs and batch-size values and found that training the model on 50 epochs with a batch size equal to 32 achieves the best results. Categorical Cross entropy is used as loss function. The learning rate of the optimizers is illustrated in table 6.

In training the CNN+SVM model, we used radial basis function (RBF) as the kernel function, C with value equal to 1 and gamma is "Scale". In the XGBoost+CNN model, deviance was used as activation function and max_depth was set to 3.

Table 6. Learning Rate Values of the Optimizers

Optimizer	Learning rate
Adam [25]	0.001
AdaDelta [26]	1.0
AdaMax [27]	0.002
AdaGrad [28]	0.01
(SGD) [29]	0.01

IV. RESULTS AND DISCUSSION

A. RESULTS

Table 7 presents the experimental results. The table consists of 30 rows that correspond to the combination of 5 optimizers and 6 weight initializers, and 6 columns that illustrate the performance of the 3 models in the 2 experiments in terms of accuracy. The labels Experiment 1 and Experiment 2 in the first row refer to the experiments that used HMB1 and HMB2, respectively. The second row in the table illustrates the Hybrid CNN models: CNN+FCL, CNN+SVM and CNN+XGBoost. The first and second columns present the optimizers and weight initializers. The third through eighth columns are the results in accuracy of the classifiers as illustrated in the second row.

Figure 7 and figure 8 show the three hybrid CNN models' testing accuracy as a line chart in Experiment 1 and Experiment 2, respectively. The rows in experiment tables are presented in the x-axis of the line chart and the y-axis refers to the classifiers' test accuracies. The blue line refers to the testing accuracy of the standalone CNN model. While the orange and green lines refer to the hybrid models that used SVM and XGBoost, respectively.

From the conducted experiments, the highest accuracy reported in Experiment 1 was 96.3% achieved by SVM, while XGBoost achieved a comparable result with the highest accuracy equal to 95.7%. The best result reported by CNN was 89.7%. The average testing accuracies of CNN, SVM and XGBoost in the same experiment are 74%, 91.4%

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

Table 7. Results of the Experiments

		Experiment 1			Experiment 2		
Optimizer	Weight Initializer	CNN	CNN + SVM	CNN+XGBoost	CNN	CNN + SVM	CNN+XGBoost
Adamax	HN	0.838	0.953	0.950	0.642	0.934	0.921
	HU	0.873	0.959	0.953	0.634	0.922	0.903
	GN	0.874	0.957	0.951	0.281	0.914	0.904
	GU	0.210	0.776	0.748	0.573	0.931	0.917
	LN	0.871	0.958	0.951	0.817	0.939	0.921
	LU	0.850	0.956	0.950	0.360	0.910	0.897
Adadelta	HN	0.888	0.962	0.955	0.738	0.941	0.925
	HU	0.897	0.962	0.957	0.608	0.898	0.876
	GN	0.666	0.940	0.932	0.647	0.928	0.917
	GU	0.886	0.962	0.954	0.435	0.934	0.921
	LN	0.882	0.963	0.957	0.587	0.934	0.921
	LU	0.826	0.951	0.946	0.069	0.829	0.822
Adam	HN	0.859	0.960	0.955	0.207	0.910	0.903
	HU	0.870	0.958	0.953	0.093	0.883	0.877
	GN	0.872	0.960	0.953	0.623	0.932	0.917
	GU	0.882	0.962	0.955	0.096	0.882	0.875
	LN	0.733	0.919	0.917	0.623	0.945	0.924
	LU	0.851	0.953	0.945	0.743	0.930	0.914
SGD	HN	0.857	0.949	0.945	0.790	0.885	0.860
	HU	0.135	0.630	0.560	0.541	0.865	0.854
	GN	0.129	0.773	0.714	0.385	0.886	0.869
	GU	0.827	0.949	0.942	0.164	0.856	0.829
	LN	0.035	0.446	0.404	0.396	0.862	0.851
	LU	0.376	0.929	0.907	0.268	0.862	0.851
Adagrad	HN	0.875	0.949	0.946	0.766	0.851	0.831
	HU	0.875	0.954	0.948	0.787	0.857	0.834
	GN	0.869	0.955	0.950	0.841	0.908	0.886
	GU	0.877	0.959	0.953	0.817	0.908	0.889
	LN	0.874	0.960	0.957	0.815	0.871	0.851
	LU	0.873	0.959	0.951	0.846	0.893	0.877
	HN = HeNormal HU = HeUniform LN = LecunNormal LU = LechunUniform GN = GlorotNormal GU = GlorotUniform						

and 90.2%, respectively. It can be noted that hybrid models that used SVM and XGBoost as classifiers outperform the original model that uses FCL.

Table 8 and table 9 represent the average and the maximum accuracies with respect to the optimizers and weight initializers in Experiment 1, respectively.

From the conducted experiments, the highest accuracy reported in Experiment 2 is 94.5%. It is achieved by the SVM classifier. The highest accuracy reported by XGBoost and CNN are 92.5% and 84.6%, respectively. The average testing accuracies of CNN, SVM and XGBoost in the same experiment are 54%, 90% and 88.5%, respectively. Table 10 and table 11 below represent the average and the maximum accuracies with respect to the optimizers and weight initializers, respectively.

Table 8. Aggregated Results Based on Optimizers in Experiment 1

Optimizer\	CNN+FCL		CNN+SVM		CNN+XGBoost	
Model	average	Max	average	max	average	max
Adamax	0.753	0.874	0.927	0.959	0.917	0.953
Adadelta	0.841	**0.897**	**0.957**	**0.963**	**0.950**	**0.957**
Adam	0.845	0.882	0.952	0.962	0.946	0.955
SGD	0.393	0.857	0.779	0.949	0.745	0.945
Adagrad	**0.874**	0.877	0.956	0.960	**0.951**	**0.957**

Table 9. Aggregated Results Based on weight initializers in Experiment1

Weight Initializers \ Model	CNN+FCL		CNN+SVM		CNN+Xgboost	
	Average	Max	Average	Max	Average	Max
HeNormal	**0.863**	0.888	**0.955**	0.962	**0.950**	0.955
He Uniform	0.730	**0.897**	0.893	0.962	0.874	**0.957**
GlorotNormal	0.682	0.874	0.917	0.960	0.900	0.953
Glorot Uniform	0.736	0.886	0.922	0.962	0.910	0.955
Lecun Normal	0.679	0.882	0.849	**0.963**	0.837	**0.957**
Lecun Uniform	0.755	0.873	0.950	0.959	0.940	0.951

Table 10. Aggregated Results Based on Optimizers in Experiment2

Optimizer\	CNN+FCL		CNN+SVM		CNN+XGBoost	
Model	average	Max	average	max	average	max
Adamax	0.551	0.817	**0.925**	0.939	**0.910**	0.921
Adadelta	0.514	0.738	0.911	0.941	0.897	**0.925**
Adam	0.398	0.743	0.914	**0.945**	0.902	0.924
SGD	0.424	0.790	0.869	0.886	0.852	0.869
Adagrad	**0.812**	**0.846**	0.881	0.908	0.862	0.889

Table 11: Aggregated Results Based on weight initializers in Experiment 2

Weight Initializers \ Model	CNN+FCL		CNN+SVM		CNN+Xgboost	
	Average	Max	Average	Max	Average	Max
HeNormal	0.629	0.790	0.904	0.941	0.888	**0.925**
He Uniform	0.533	0.787	0.885	0.922	0.869	0.903
GlorotNormal	0.555	0.841	**0.914**	0.932	**0.899**	0.917
Glorot Uniform	0.417	0.817	0.902	0.934	0.886	0.921
Lecun Normal	**0.648**	0.817	0.910	**0.945**	0.894	0.924
Lecun Uniform	0.457	**0.846**	0.885	0.930	0.872	0.914

B. Discussion

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

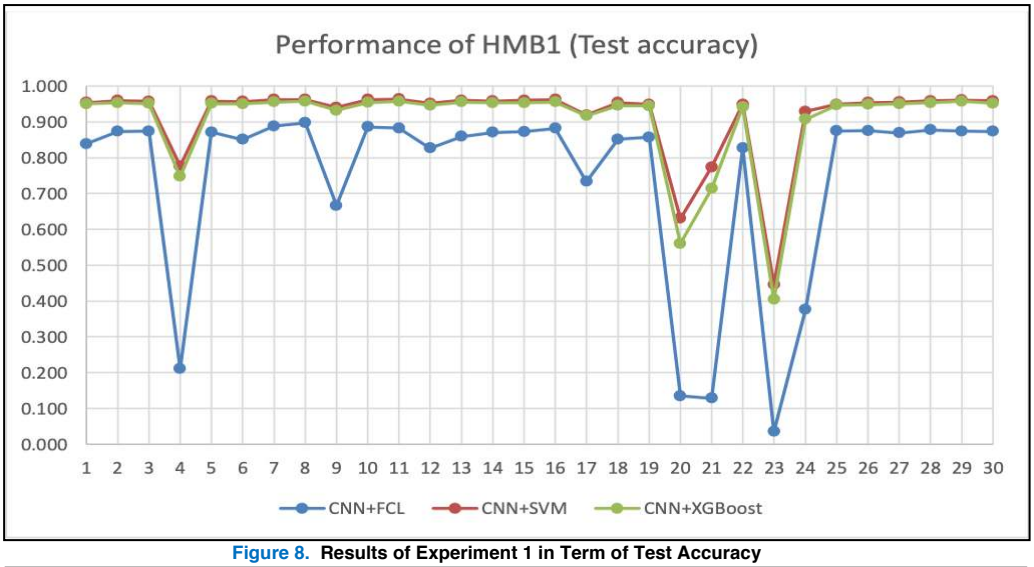

Figure 8. Results of Experiment 1 in Term of Test Accuracy

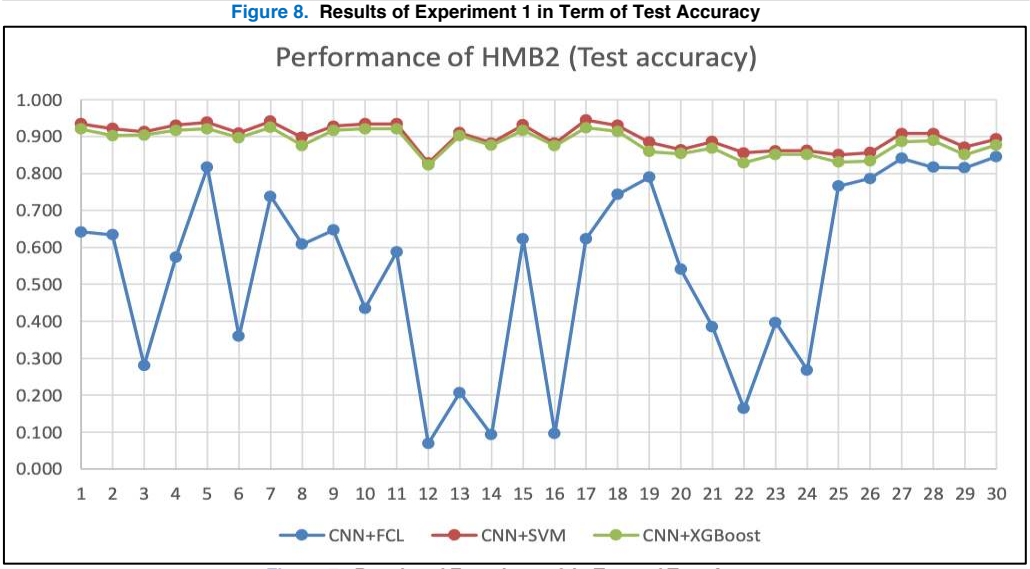

Figure 7. Results of Experiment 2 in Term of Test Accuracy

The main objective of the research is to build a hybrid model that delivers better performance in recognizing Arabic Handwritten Characters. In this section, the performance of the hybrid CNN-based architectures will be discussed according to three aspects: the base CNN architectures, the ML classifiers used in the hybrid models, and optimizers and weight initializers.

Hijja was the dataset used to evaluate the hybrid models. As mentioned in [1], the performance of different CNN-based models produced significantly lower results in the Hijja compared to other Arabic Handwritten Characters datasets. Therefore, it is considered a more complex and difficult dataset. To the best of our knowledge, the highest accuracy achieved in Hijja dataset was 88%. The base CNN models of HMB1 and HMB2 with softmax FCL as output layer achieved very close results to the Hijja's best results with accuracies of 89% and 87%, respectively. The results of the HMB1 are better than HMB2 in all three classifiers. However, HMB1 is a more complex architecture with 6,071,693 trainable parameters compared to HMB2's179,869 trainable parameters.

From the results of Experiment 1 and Experiment 2, the best results in terms of test accuracy for FCL, SVM and XGBoost as classifiers were 89%, 96.3% and 95.7%, respectively. While the lowest testing accuracies for FCL, SVM and XGBoost were 3.5%, 44.6% and 40.4%, respectively. It can be concluded that the performance of SVM and XGBoost as classifiers in hybrid models outperforms the FCL as classifier by a large margin. From tables 8, 9, 10 and 11, SVM achieved slightly better results than XGBoost.

In general, Adadelta achieved the highest accuracy in HMB1 with the three classifiers and Adam in HMB2. HeNormal achieved the best aggregated accuracy in HMB1, while GlorotNormal achieved best aggregated accuracies in both models.

Poor performance of the CNN model with FCL can be justified in that CNN as a whole requires a larger dataset when compared to other algorithms like SVM and XGBoost.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access

Zhang et al. [37] stated that in case of a simple 2-layer CNN, a 2n+d number of parameters are sufficient to classify a dataset with n number of samples with d dimensions. Implementing this approach to our dataset, the number of parameters of CNN models were 6,071,693 and 179,869 for HMB1 and HMB2 respectively. Our dataset had 8,174 training examples in 29 classes. Thus, 2*8,174+29 = 16,377 is quite small in comparison with this high number of parameters. This verifies that our dataset was too small to be classified by a classic CNN. In comparison, SVM and XGBoost have no such limitation of a large dataset.

The only thing that has remained same in our classification model is the CNN architecture for feature extraction. The things that have changed are FCL, SVM and XGBoost at classification stage. When the comparison of parameters is done for these architectures, it can be seen that there is a huge difference of parameters. For FCL the total number of parameters depends on number of layers and neurons, these parameters are weights and biases that needs to be tuned. Total number of trainable parameters in FCL that we used are 698,653. Which is still a large number in comparison with 16,377. In contrast this is not the case for SVM and XGBoost. SVM and XGBoost have a limited number of parameters when compared with CNN (including FCL). [38, 39]. The number of parameters to be tuned in SVM and XGBoost can not be calculated accurately as many factors involved in them are dynamic i.e. number of trees, depth of trees. Despite of it, we can still make a comparison. Only a few parameters are needed to be tuned in case of SVM and XGBoost [38, 39], for SVM these parameters are to tune the parameter values of the kernel that is being used for classification i.e. linear, radial basis function (RBF) etc. and in XGBoost these parameters depends on booster that has been selected and learning task that needs to be done i.e. regression or tree. Regression has a few parameters to be tuned depending upon number of variables in the dataset but in case of tree the number of parameters to be tuned are the depth of the tree of the number of the trees to be used. Which in comparison are still smaller than the number of parameters in FCL.

C. Comparison

An explained literature review of how much work is present in the field of Arabic character recognition. Table 1 shows the work done using CNN models and Table 2 shows the work done on Arabic character recognition using hybrid models. In case of using normal CNN models 6 different types of datasets has been used which are AHCD, HMBD, AIA9k, OIHAC, HACDB and Hijja. The dataset used in abundance and in almost every study was AHCD [11], for which a maximum accuracy was obtained of 99.98% by [7] by using AlexNet.

In the case of hybrid models, HACDB and AHCD was used. Accuracy achieved by using AHCD in its original paper was 94.90%, which was improved with hybrid model [18], which achieved 95.07% accuracy.

We compare our methodology and the one that was conducted by the creators of the Hijja dataset to see the differences and effectiveness in how both models performed. The Hijja dataset is new so there is not much work present which uses it, [4] used the Hijja dataset and achieved an accuracy of 88%, which we improved by using hybrid models. The maximum accuracy we achieved was 96.3% by using SVM at classification stage.

V. CONCLUSION

The original Hijja experiment designed a convolutional neural network (CNN) and used it for both feature extraction and classification of the Arabic characters. It ended up with its best accuracy being 88%. But the same CNN was used to experiment on the AHCD dataset and it achieved an accuracy of 97%, which shows that at that time, the Hijja dataset was imbalanced and needed to be reevaluated.

In our experiment, we analyzed the Hijja dataset and found irregularities, like some blurry letters and some distorted and unclear symbols. We combined an ML model and Deep Learning model to form a hybrid model, since CNNs are great for feature extraction and ML models are great at classification, implementing a combination of the two with each one serving the purpose it is best at, giving us an effective hybrid model. We conducted two experiments and attained an accuracy of 96.3%, which is 8% higher than the original Hijja experiment. Our hybrid model even achieved an accuracy comparable to the one conducted on the AHCD dataset using the Hijja model, which shows that our hybrid model is highly effective.

Acknowledgment: We would like to thank the Deanship of Scientific Research, Qassim University for funding the publication of this project.

REFERENCES

[1] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Comput. Appl.*, pp. 1–13, Jun. 2020, doi: 10.1007/s00521-020-05070-8.

[2] H. T. Weldegebriel, H. Liu, A. U. Haq, E. Bugingo, and D. Zhang, "A New Hybrid Convolutional Neural Network and eXtreme Gradient Boosting Classifier for Recognizing Handwritten Ethiopian Characters," *IEEE Access*, vol. 8, pp. 17804–17818, 2020, doi: 10.1109/ACCESS.2019.2960161.

[3] "Babbel Magazine." [Online]. Available: https://www.babbel.com/en/magazine/ (accessed Mar. 09, 2021).

[4] U. Porwal, Z. Shi, and S. Setlur, "Machine learning in handwritten arabic text recognition," in *Handbook of Statistics*, vol. 31, Elsevier B.V., 2013, pp. 443–469.

[5] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten Arabic character recognition," *Cogn. Syst. Res.*, vol. 50, pp. 180–195, Aug. 2018, doi: 10.1016/j.cogsys.2017.11.002.

[6] I. P. de Sousa, "Convolutional ensembles for Arabic Handwritten Character and Digit Recognition," *PeerJ Comput. Sci.*, vol. 2018, no. 10, p. e167, Oct. 2018, doi: 10.7717/peerj-cs.167.

[7] M. Elleuch, H. Lahiani, and M. Kherallah, "Recognizing Arabic Handwritten Script using Support Vector Machine classifier," in *International Conference on Intelligent Systems Design and Applications, ISDA*, Jun. 2016, vol. 2016-June, pp. 551–556, doi: 10.1109/ISDA.2015.7489176.

[8] M. A. Mudhsh and R. Almodfer, "Arabic Handwritten Alphanumeric Character Recognition Using Very Deep Neural Network," *Information*, vol. 8, no. 3, p. 105, Aug. 2017, doi: 10.3390/info8030105.

[9] 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, Apr. 2012, doi: 10.1016/j.patcog.2011.09.021.

[10] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata, "Pedestrian Detection with Convolutional Neural Networks Feature Gray-scale value Haar-wavelet coefficients Four directional features Vertical and horizontal edge intensities Edge intensities Gradient image Rectangular filter Chamfer," in *IEEE Intelligent Vehicles Symposium*, 2005, pp. 224–229.

[11] A. El-Sawy, M. Loey, and H. El-Bakry, "Arabic Handwritten Characters Recognition using Convolutional Neural Network," *WSEAS Trans. Comput. Res.*, vol. 5, no. 1, pp. 11–19, 2017.

[12] H. M. Balaha, H. A. Ali, and M. Badawy, "Automatic recognition of handwritten Arabic characters: a comprehensive review," *Neural Computing and Applications*. Springer, pp. 1–24, Jul. 17, 2020, doi: 10.1007/s00521-020-05137-6.

[13] K. Simonyan and A. Zisserman, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION," 2015. Accessed: Nov. 15, 2020. [Online]. Available: http://www.robots.ox.ac.uk/.

[14] M. Torki, M. E. Hussein, A. Elsallamy, M. Fayyaz, and S. Yaser, "WINDOW-BASED DESCRIPTORS FOR ARABIC HANDWRITTEN ALPHABET RECOGNITION: A COMPARATIVE STUDY ON A NOVEL DATASET."2014, *arXiv: 1411.3519* [Online]. Available: http://arxiv.org/abs/1411.3519

[15] H. Alyahya, M. M. Ben Ismail, and A. Al-Salman, "Deep ensemble neural networks for recognizing isolated Arabic handwritten characters," *Accent. Trans. Image Process. Comput. Vis.*, vol. 6, no. 21, pp. 68–79, Nov. 2020, doi: 10.19101/tipcv.2020.618051.

[16] M. Kherallah, M. Elleuch, and N. Tagougui, "A novel architecture of CNN based on SVM classifier for recognising Arabic handwritten script," *Int. J. Intell. Syst. Technol. Appl.*, vol. 15, no. 4, p. 323, 2016, doi: 10.1504/ijista.2016.10000779.

[17] M. Elleuch, R. Maalej, and M. Kherallah, "A New design based-SVM of the CNN classifier architecture with dropout for offline Arabic handwritten recognition," in *Procedia Computer Science*, Jan. 2016, vol. 80, pp. 1712–1723, doi: 10.1016/j.procs.2016.05.512.

[18] M. Shams, A. A., and W. Z., "Arabic Handwritten Character Recognition based on Convolution Neural Networks and Support Vector Machine," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 8, pp. 144–149, 2020, doi: 10.14569/IJACSA.2020.0110819.

[19] H. M. Balaha, H. A. Ali, M. Saraya, and M. Badawy, "A new Arabic handwritten character recognition deep learning system (AHCR-DLS)," *Neural Comput. Appl.*, pp. 1–43, Oct. 2020, doi: 10.1007/s00521-020-05397-2.

[20] A. El Sawy, H. El-Bakry, and M. Loey, "Arabic Handwritten Characters Dataset (AHCD)," 2017. Accessed: Jan. 18, 2021. [Online]. Available: https://www.kaggle.com/mloey1/ahcd1.

[21] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997, doi: 10.1006/jcss.1997.1504.

[22] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001, doi: 10.1214/aos/1013203451.

[23] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, Jun. 1990, doi: 10.1007/bf00116037.

[24] T. Chen and T. He, "xgboost: eXtreme Gradient Boosting," 2021. R package version. 0.4-4, 2016, [online] Available: https://CRAN.R-project.org/package=xgboost.

[25] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," Dec. 2015, *arXiv: 1412.6980v9*. [Online]. Available: https://arxiv.org/abs/1412.6980v9.

[26] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," Dec. 2012, *arXiv: 1212.5701*. [Online]. Available: http://arxiv.org/abs/1212.5701.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016, *arXiv: 1609.04747*. [Online]. Available: http://arxiv.org/abs/1609.04747.

[28] J. Duchi, E. Hazan and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *J. Mach. Learn. Res.*, vol. 12, pp. 2121-2159, Jul. 2011.

[29] L. Bottou, "Stochastic gradient descent tricks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7700 LECTURE NO, pp. 421–436, 2012, doi: 10.1007/978-3-642-35289-8_25.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," JMLR Workshop and Conference Proceedings, Mar. 2010. Accessed: Mar. 02, 2021. [Online]. Available: http://www.iro.umontreal.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034, doi: https://doi.org/10.1109/ICCV.2015.123.

[32] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7700 LECTURE NO, pp. 9–48, 2012, doi: 10.1007/978-3-642-35289-8_3.

[33] "Welcome To Colaboratory - Colaboratory." Accessed:Mar. 01, 2021 [online] Available: https://colab.research.google.com/notebooks/intro.ipynb.

[34] Giancarlo Zaccone, *Getting Started with TensorFlow* -. Packt Publishing, 2016.

[35] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *Proceedings of the International Joint Conference on Neural Networks*, Sep. 2015, vol. 2015-September, doi: 10.1109/IJCNN.2015.7280578.

[36] R. A. Dunne and N. A. Campbell, "On The Pairing Of The Softmax Activation And Cross Entropy Penalty Functions And The Derivation Of The Softmax Activation Function," in *Conference on Neural Networks*, 1997, pp. 181–185.

[37] C. Zhang, S. Bengio, M. Hardt, B.Recht and O. Vinyals, "Understanding deep learning requires rethinking generalization," Nov. 2016, *arXiv:1611.03530* [Online]. Available: http://arxiv.org/abs/1611.03530

[38] S. Thongsuwan, S. Jaiyen, A. Padcharoen, and P. Agarwal, "ConvXGB: A new deep learning model for classification problems based on CNN and XGBoost," Nuclear Engineering and Technology 53, no. 2 (2021): 522-531.

[39] Hasan, Hayder, Helmi ZM Shafri, and Mohammed Habshi. "A Comparison Between Support Vector Machine (SVM) and Convolutional Neural Network (CNN) Models For Hyperspectral Image Classification." In IOP Conference Series: Earth and Environmental Science, vol. 357, no. 1, p. 012035. IOP Publishing, 2019.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3087647, IEEE Access