Information Sciences with Applications, Vol. 3, 2024
https://doi.org/10.61356/j.iswa.2024.3294

# Advancing Arabic Handwriting Recognition with Convolutional and Recurrent Neural Network Ensembles

**Mohamed G. Mahdi [1,2,*], Ahmed Sleem [3], Ibrahim Elhenawy [2] and Soha Safwat [4]**

[1] Department of Computer Science, Higher Institute for Computer Sciences and Information Systems, 5th Settlement, New Cairo, Egypt; mohamed.grisha@cis.edu.eg.
[2] Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah, Egypt; Emails: m.gresha@fci.zu.edu.eg; ielhenawy@zu.edu.eg.
[3] Department of Computer Science, Faculty of Computers and Informatics, Tanta University, Tanta, Egypt; ahmed.selim@ics.tanta.edu.eg.
[4] Department of Computer Science, Faculty of Computers and Information Systems, Egyptian Chinese University, Cairo, Egypt; ssafwat@ecu.edu.eg.

* Correspondence: mohamed.grisha@cis.edu.eg.

**Abstract:** In recent years, Deep learning has shown significant success in English character recognition due to the language's global prevalence. Arab researchers are adapting modern English language processing advancements to Arabic, Urdu, and Pashto to align with global trends. Applying machine learning and deep learning to Arabic and similar languages presents challenges, with researchers striving to match English language recognition results. Therefore, in this research, the most popular methods, techniques, and technologies in deep learning, such as CNN, LSTM, Bi-LSTM, GRU, and Bi-GRU were used to create new models to enhance performance when applied and tested on Arabic language datasets: AHCD, and Hijjaa. Subsequently, a comparison was made between the results obtained from these different techniques using performance measurement methods such as Precision, Recall, F1-Score, and Accuracy. The comparison revealed that Bi-GRU achieved the highest performance in the AHCD dataset with an accuracy rate of 95.7%, while CNN achieved the highest performance in the Hijjaa dataset with an accuracy rate of 86.3 %.

**Keywords:** Arabic Natural Language Processing; Optical Character Recognition; Handwritten Character Recognition; Deep Learning; CNN.

## 1. Introduction

Arabic is a widely spoken Semitic language with around 274 million speakers as of 2023 to the Statista website1 shown in Figure 1. It is the official language in 25 countries and holds significance for Muslims as the language of the Holy Quran. Arabic writing is cursive, read from right to left, with 28 characters and various diacritical marks. Translating English to Arabic can be difficult due to multiple meanings and other languages borrowing from Arabic, presenting challenges for researchers in Arabic language comprehension [1–4].

Automatic recognition methods for handwritten data are crucial due to many individuals still resort to handwriting despite technological advancements [5]. These methods facilitate the offline processing of scanned handwritten documents or online entry of handwriting data, enabling the conversion of handwritten information into digital representations [3,6,7]. Developing automatic handwriting recognition systems is challenging due to variations in handwriting and unique language features, especially in handwritten character recognition within document image

[1] https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/

processing [7]. While languages like English, French, and Chinese have been extensively researched, others like Arabic have received less attention [6].

In recent times, the field of handwritten Arabic character recognition has garnered significant attention in research circles. This surge in interest can be attributed to the vital role played by the Arabic language, which ranks among the top five most widely spoken languages globally and serves as a medium of communication for hundreds of millions of individuals from various countries [8]. On the other hand, in the fields of pattern recognition and computer vision, handwritten Arabic recognition of characters is a difficult issue. It takes a lot of work to create generalized systems that can handle a variety of recognizing challenges and achieve high accuracy [9]. These difficulties result from the unique qualities of Arabic script, which include its cursive style, the inclusion of dots and diacritical marks, diagonal strokes, changes in character forms within words, and a host of other elements [3,6,8]. Furthermore, individuals' handwriting styles vary greatly from one another, and even within the same person, there may be significant or slight variations in handwriting on different occasions. These variations make it challenging for a recognition system to accurately interpret the letters in an individual's handwriting [7].

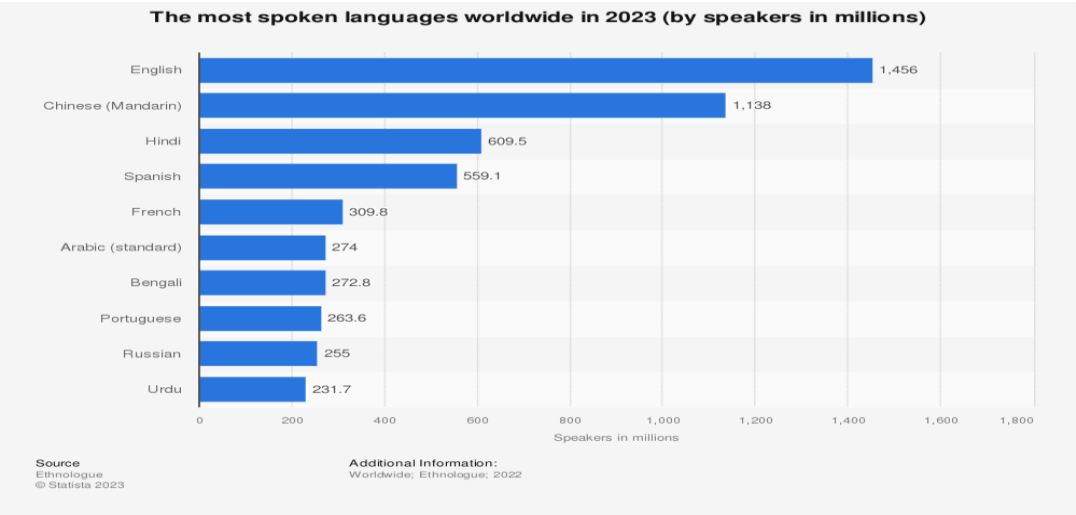



**Figure 1.** The most spoken languages worldwide in 2023.

Because studies have shown that adult handwriting can achieve remarkable accuracy rates of up to 99% when deep learning and machine learning techniques are applied, adult handwriting has been the focus of most research related to handwritten Arabic character recognition [10–17]. However, there has been a recent shift in focus among a small number of researchers towards children's handwriting data for Arabic letter recognition, given its significant importance in various applications and contexts [7,18–22]. Future research disciplines will find it necessary to use character recognition skills in applications related to children, such as education [23], interactive learning, physical or mental health assessment, and other potential practical applications. Nevertheless, this endeavor presents an additional challenge due to the notable dissimilarities between the characteristics of adult and kid handwriting in several areas, such as more variance, more noticeable deformities, and overall inferior quality [18].

Significant progress has been made in handwritten Arabic character recognition technology, with significant progress achieved through the utilization of various methods including artificial neural networks (ANNs), k-nearest neighbor (KNN), support vector machines (SVMs), and most recently, convolutional neural networks (CNNs). CNNs have emerged as a superior choice, surpassing traditional machine learning (ML) techniques that rely on manually engineered features. CNNs possess the ability to automatically identify and identify and extract standout characteristics

from the input pictures [21]. Moreover, the construction of hybrid systems for handwriting recognition, combining CNNs as feature extractors, and Using machine learning techniques as classifiers, several datasets of handwritten Arabic letters have shown positive results [17,21].

The main contributions of this work are as follows:

- Study and understanding of the various techniques used in Deep learning.
- Study and understanding of the advancements made by others in the field of Arabic handwriting recognition.
- Application of different techniques in Deep learning to Arabic handwriting recognition.
- Comparison of our findings with previous studies (state-of-the-art models).

This paper is structured as follows: Related work is presented in Section 2; the research methodology is presented in Section 3; Experiments are presented in Section 4; the results are discussed in Section 5; and lastly, the study is concluded in Section 6.

## 2. Related Work

Within this section, a comprehensive review of the existing literature is conducted, examining a range of methodologies that employ machine learning and deep learning techniques for the recognition of handwritten Arabic characters in both adults and children. The majority of recent studies that are pertinent to our research primarily concentrate on introducing diverse approaches to address this demanding task, with a particular emphasis on utilizing models based on Convolutional Neural Networks (CNNs).

In 2017, El-Sawy et al. [10] proposed an innovative CNN model that was trained and tested on their proprietary dataset known as AHCD. This dataset consists of 16,800 handwritten Arabic characters collected from 60 individuals aged between 19 and 40 years, categorized into 28 classes. Their model achieved an impressive accuracy of 94.9% on this dataset.

In a separate study conducted in 2017 by Younis [24], a deep model utilizing CNNs was proposed for the recognition of handwritten Arabic letters. To mitigate overfitting, the researchers employed multiple optimization strategies. The outcomes revealed that their model successfully classified letters using two distinct datasets: AIA9k and AHCD. The achieved accuracies were 94.8% for AIA9k and 97.6% for AHCD.

De Sousa [12] conducted a study introducing two deep models, namely VGG12 and REGU, to recognize handwritten Arabic letters and numbers. These models underwent training using two different approaches: one with data augmentation and the other without. Following this, an ensemble model was created by combining the predictions from all four models. The ensemble model demonstrated exceptional performance, achieving the highest accuracy of 98.42% for the AHCD dataset and 99.47% for the MADbase dataset.

Boufenar et al. [13] conducted a study where they devised a DCNN model based on the architecture of AlexNet. Their primary focus was on investigating the influence of preprocessing data samples in improving the model's performance. Three different learning strategies were explored: training the model from scratch, utilizing transfer learning, and fine-tuning the CNN. The experimental findings consistently demonstrated that the first approach yielded superior results compared to the other two, irrespective of whether preprocessing techniques were applied or not. Remarkably, the model achieved an average accuracy of 100% on the OIHACDB-40 dataset and 99.98% on the AHCD dataset.

In a separate study by Alyahya et al. [15], It was investigated if handwritten Arabic letters might be recognized by the ResNet-18 architecture. The study presented four ensemble models, which comprised the original ResNet-18 model and an enhanced version with an additional fully connected layer, with or without a dropout layer. Furthermore, two models with two fully connected layers,

with or without dropout, were also considered. Among these ensemble models, the original ResNet-18 model demonstrated superior performance, achieving the highest test score of 98.30% on the AHCD dataset.

In another study by Alkhateeb et al. [18], A system based on deep learning was introduced to identify handwritten Arabic letters. The system utilized CNN and was evaluated on three separate datasets: AHCR, AHCD, and Hijja. The proposed method yielded accuracy rates of 89.8%, 95.4%, and 92.5% on the AHCR, AHCD, and Hijja datasets, correspondingly.

In [25], a CNN model was developed for the recognition of handwritten Arabic letters. The model was trained and tested using the AHCD dataset. The experiment demonstrated that the proposed method achieved a recognition rate of 97.2%. Notably, when data augmentation techniques were applied, the model's accuracy increased to 97.7%.

In a study conducted by Altwaijry et al. [7] in 2020, the focus shifted towards recognizing Arabic letters in children's handwriting. With 47,434 distinct and linked Arabic characters written by youngsters between the ages of 7 and 12, the researchers created a special dataset they named Hijja. They also developed a CNN-based model to assess its performance on their dataset. A comparison was made with the model proposed by El-Sawy [10] on both the Hijja and AHCD datasets. The experimental findings demonstrated that their model surpassed the compared model, attaining accuracy rates of 88% and 97% on the Hijja and AHCD datasets, respectively.

Taking a different approach, Alrobah et al. [21] combined CNN deep-learning models for feature extraction with SVM and XGBoost machine-learning models for classification, creating a hybrid model. This hybrid model demonstrated efficiency by achieving an accuracy of 96.3% on the Hijja dataset using the HMB1 model and the SVM classifier.

Furthermore, Ullah et al. [14] investigated the impact of the dropout technique on their CNN model. They observed a significant difference in performance when training the model with and without dropout, demonstrating that dropout regularization effectively mitigates overfitting. The model achieved a test accuracy of 96.78% on the AHCD dataset when dropout was applied.

In another study by Ali et al. [17], a CNN-based SVM model with dropout was designed for recognizing handwritten Arabic letters. The model utilized two deep neural networks and was evaluated on various datasets, including AHDB, AHCD, HACDB, and IFN/ENIT. The authors reported notable performance improvements compared to previous models, achieving accuracy rates of 99%, 99.71%, 99.85%, and 98.58% on AHDB, AHCD, HACDB, and IFN/ENIT, respectively.

Nayef et al. [19] presented a study focusing on CNN models for recognizing handwrit-ten Arabic characters while incorporating an improved Leaky-ReLU activation function. Four datasets were used to evaluate their models: AHCD, HIJJA, MNIST, and their dataset containing 38,100 handwritten Arabic characters. The proposed CNN model, employing Leaky-ReLU optimization, surpassed the model mentioned in [24], achieving accuracy rates of 99%, 95%, and 90% on AHCD, the researchers' dataset, and Hijja, respectively.

In 2022, Wagaa et al. [20], a new CNN architecture was introduced, achieved accuracy rates of 98.48% and 91.24% on the AHCD and Hijja datasets, respectively. They incorporated rotation and shifting data augmentation techniques and utilized the Nadam optimizer. The researchers also explored the impact of combining the AHCD and Hijja datasets in varying proportions during training and testing, employing different data augmentation approaches. Their findings demonstrated that utilizing the Nadam optimizer in conjunction with rotation and shifting data augmentation techniques resulted in the highest test accuracy of 98.32% when combining 80% of AHCD and 20% of Hijja for training, along with 20% of AHCD and 10% of Hijja for testing.

Bouchriha et al. [22] presented a novel CNN model for recognizing handwritten Arabic characters, specifically focusing on the unique characteristics of Arabic text, such as the variation in letter shapes based on their position within a word. Using the Hijja dataset, they achieved an accuracy

of 95%. A summary of related work on handwritten Arabic character recognition is shown in Table 1.

**Table 1.** A summary of related work on handwritten Arabic character recognition.

| Reference | Feature Extractor | Classifier | Dataset | Size | Accuracy |
|---|---|---|---|---|---|
| **El-Sawy et al., 2017[10]** | CNN | SoftMax | AHCD | 16,800 | 94.9% |
| **Younis, 2017[24]** | CNN | SoftMax | AHCD | 16,800 | 97.6% |
| **de Sousa, 2018[12]** | CNN | SoftMax | AHCD | 16,800 | 98.42% |
| **Boufenar et al., 2018[13]** | CNN | SoftMax | AHCD | 16,800 | 99.98% |
| **Alyahya et al., 2020[15]** | CNN | SoftMax | AHCD | 16,800 | 98.3% |
| **Alkhateeb, 2020[18]** | CNN | SoftMax | Hijja<br>AHCD | 47,434<br>16,800 | 92.5%<br>95.4% |
| **AlJarrah et al., 2021[25]** | CNN | SoftMax | AHCD | 16,800 | 97.7% |
| **Altwaijry & Al-Turaiki, 2021a[7]** | CNN | SoftMax | Hijja<br>AHCD | 47,434<br>16,800 | 88%<br>97% |
| **Alrobah & Albahli, 2021[21]** | CNN | SoftMax<br>SVM<br>XGBoost | Hijja | 47,434 | 89%<br>96.3%<br>95.7% |
| **Ullah & Jamjoom, 2022[14]** | CNN | SoftMax | AHCD | 16,800 | 96.78% |
| **Ali & Mallaiah, 2022[17]** | CNN | SVM | AHCD<br>HACDB | 16,800<br>6600 | 99.71%<br>99.85% |
| **Nayef et al., 2022[19]** | CNN | SoftMax | AHCD<br>Hijja | 16,800<br>47,434 | 99%<br>90% |
| **Wagaa et al., 2022[20]** | CNN | SoftMax | AHCD<br>Hijja | 16,800<br>47,434 | 98.48%<br>95% |
| **Bouchriha et al., 2022[22]** | CNN | SoftMax | Hijja | 47,434 | 95% |

## 3. Materials and Methods

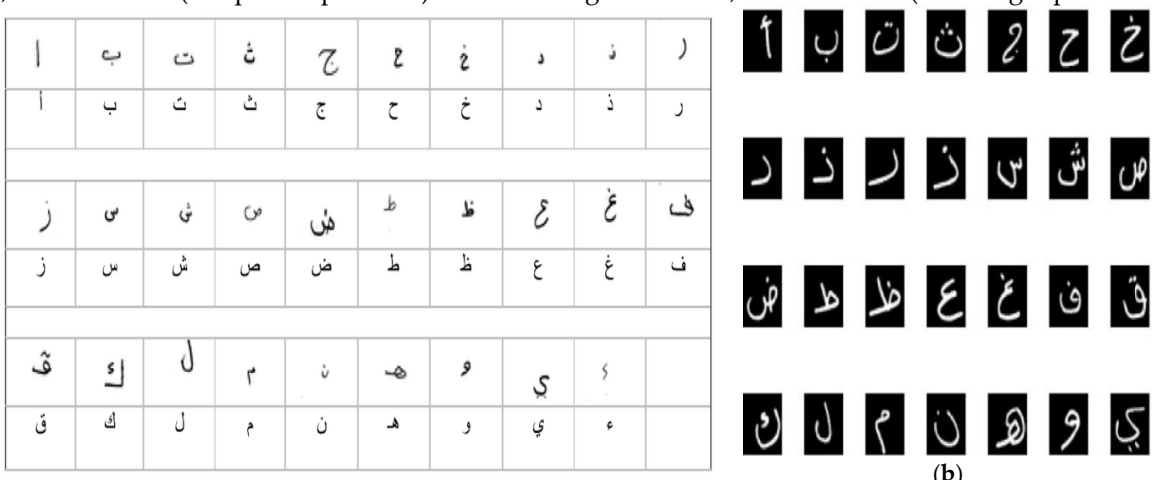
In this research, we will employ the most utilized strategies and techniques in deep learning, specifically in the field of NLP, for Arabic Recognition of Handwritten Characters and numbers. We will present and discuss these strategies, methods, and their implementation plan in the research, supported by links on GitHub and Kaggle. This is aimed at sharing our findings with researchers worldwide who are interested in this specialization, to support scientific collaboration and validate the results of our research, ensuring that no one can claim the results are inaccurate.

### *3.1 Datasets Description*

For all experiments conducted in this paper, we utilized publicly available datasets comprising of Hijjaa Data set, and the Arabic Handwritten Characters Data set (AHCD).

The Hijjaa dataset is a publicly accessible collection of individual Arabic letters that was released recently. It was first presented by Altwaijry et al. [7]. It was made by seven to twelve-year-old Saudi Arabian schoolchildren in Riyadh. There are 108 classes in this dataset, each of which corresponds to a different Arabic letter. The letters are shown in four different configurations: alone, at the start, middle, and finish of a word. The Hijjaa dataset has 47,434 photos in total. These pictures are arranged into 29 different files, each of which has pictures for a different Arabic letter. There is also a single file devoted to the letter from Hamza.

An enormous collection of handwritten Arabic script characters is called the Arabic Handwritten Characters Dataset (AHCD) [10]. A large variety of Arabic characters, both standalone and contained in words, are covered by this dataset. It provides a wide range of writing styles and variants and does a good job of encapsulating the subtleties of Arabic handwriting. The dataset, which has 16,800 characters overall, was produced by 60 individuals who were between the ages of 19 and 40. Ninety percent of the participants are right-handed people. All characters ('alef' to 'yeh') were written by each participant 10 times in two different formats. Two separate sets comprise the dataset: a test set with 3,360 characters (120 photos per class) and a training set with 13,440 characters (480 images per class).



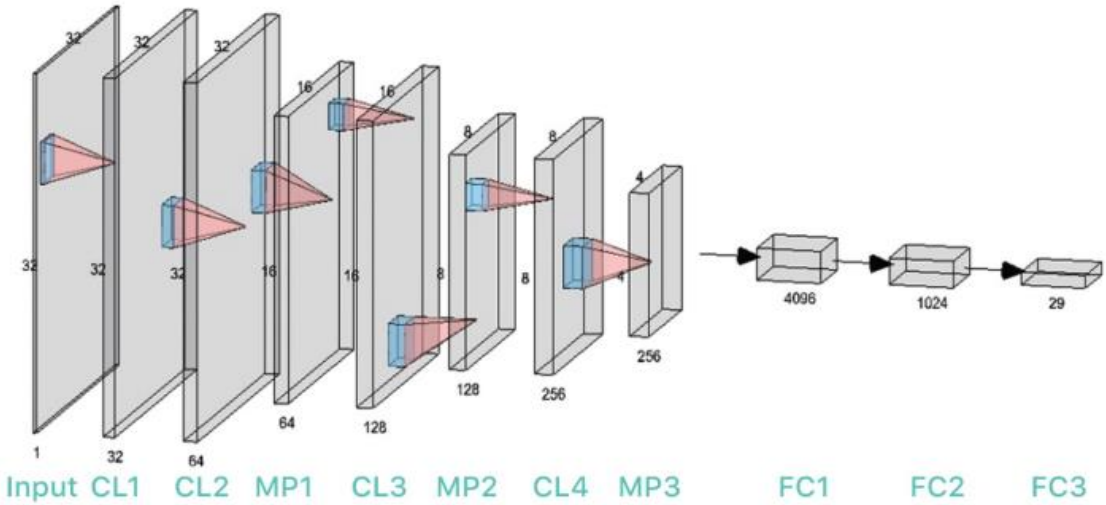
(**a**) (**b**)

**Figure 2.** This is a figure, an example of datasets (a) Description of the sample of the Hijja dataset.; (b) Description of the sample of the AHCD dataset.

### *3.2 Proposed Models*

#### 3.2.1 Convolutional Neural Network (CNN)

In this section, we will rebuild the model described in [23] to ensure the result and compare the result with other models described below. The CNN deep learning model proposed consisted of a total of seven layers, as illustrated in Figures 3(a), and 3(b). The architecture of the model consisted of four initial convolutional layers, followed by three fully connected layers. Between the convolutional and fully connected layers, there were pooling and activation layers. The ReLU nonlinearity function was utilized in the hidden layers to enhance the efficiency of the training phase.




(**a**)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 64)        640

 conv2d_1 (Conv2D)           (None, 28, 28, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 27, 27, 64)       0
 )

 batch_normalization (BatchN  (None, 27, 27, 64)       256
 ormalization)

 conv2d_2 (Conv2D)           (None, 25, 25, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 24, 24, 128)      0
 2D)

 batch_normalization_1 (Batc  (None, 24, 24, 128)      512
 hNormalization)

 conv2d_3 (Conv2D)           (None, 22, 22, 128)       147584

 max_pooling2d_2 (MaxPooling  (None, 21, 21, 128)      0
 2D)

 batch_normalization_2 (Batc  (None, 21, 21, 128)      512
 hNormalization)

 dropout (Dropout)           (None, 21, 21, 128)       0

 flatten (Flatten)           (None, 56448)             0

 dense (Dense)               (None, 512)               28901888

 dense_1 (Dense)             (None, 1024)              525312

 dense_2 (Dense)             (None, 29)                29725

=================================================================
Total params: 29,717,213
Trainable params: 29,716,573
Non-trainable params: 640
_________________________________________________________________
```

(**b**)

**Figure 3.** (**a**) Description of CNN architecture.; (**b**) Description of CNN implementation.

### 3.2.2 Long Short-Term Memory (LSTM)

The model architecture consists of multiple layers connected sequentially. The input layer is followed by an LSTM layer with 256 units, which is connected to a fully connected layer with 256 units and ReLU activation. The output of this layer is then fed into another LSTM layer with 128 units, which is connected to a fully connected layer with 128 units and ReLU activation. The process is repeated with two more LSTM layers, one with 64 units and the other with 32 units, both connected to their respective fully connected layers with ReLU activation. Following the LSTM layers, there is an attention layer, which is then followed by a flattened layer to reshape the data. The flattened output is then passed through two fully connected layers with 128 and 256 units, respectively, both using ReLU activation. Finally, the last fully connected layer with the number of units equal to the total number of classes in the classification task is added, using SoftMax activation to provide the final classification probabilities. Figure 4(a) shows the layers and the output shape of each parameter.

### 3.2.3 Gated Recurrent Unit (GRU)

The model architecture consists of multiple layers connected sequentially. The input shape is (32, 32), representing sequences of length 32 with each element having a dimension of 32. The model starts with a GRU layer with 256 units and a dropout of 0.25. The output from this layer is then passed to a Dense layer with 256 units and ReLU activation. The process is repeated with two more GRU layers, each followed by a Dense layer with ReLU activation. After the last GRU layer, the data flows into a flattened layer, which reshapes the data from a 2D shape to a 1D shape. The flattened output is then passed through three Dense layers with 256, 128, and 64 units, respectively, all using ReLU activation. Finally, the last Dense layer is added with the number of units equal to the total number

of classes in the classification problem. It uses SoftMax activation to provide the final classification probabilities. Figure 4(b) shows the layers and the output shape of each parameter.

### 3.2.4 Bidirectional Long Short-Term Memory (Bi-LSTM)

The model architecture consists of multiple layers connected sequentially. The input shape is (32, 32), representing sequences of length 32 with each element having a dimension of 32. The model starts with an Input Layer that takes the specified input shape. It is followed by a Bidirectional LSTM layer with 256 units and a dropout of 0.25. The output from this layer is then passed to a Dense layer with 256 units and ReLU activation. The process is repeated with three more Bidirectional LSTM layers, each followed by a Dense layer with ReLU activation. After the last Bidirectional LSTM layer, the data flows into a flattened layer, which reshapes the data from a 2D shape to a 1D shape. The flattened output is then passed through two Dense layers with 128 and 256 units, respectively, both using ReLU activation. Finally, the last Dense layer is added with the number of units equal to the total number of classes in the classification problem. It uses SoftMax activation to provide the final classification probabilities. Figure 4(c) shows the layers and the output shape of each parameter

### 3.2.5 Bidirectional Gated Recurrent Unit (Bi-GRU)

The model architecture consists of multiple layers connected sequentially. The input shape is (32, 32), representing sequences of length 32 with each element having a dimension of 32. The model starts with a Bidirectional GRU layer with 256 units and a dropout of 0.25. The output from this layer is then passed to a Dense layer with 256 units and ReLU activation. The process is repeated with three more Bidirectional GRU layers, each followed by a Dense layer with ReLU activation. After the last Bidirectional GRU layer, the data flows into a flattened layer, which reshapes the data from a 2D shape to a 1D shape. The flattened output is then passed through two Dense layers with 128 and 256 units, respectively, both using ReLU activation. Finally, the last Dense layer is added with the number of units equal to the total number of classes in the classification problem. It uses SoftMax activation to provide the final classification probabilities. Figure 4(d) shows the layers and the output shape of each parameter.

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| lstm (LSTM) | (None, 32, 256) | 295936 |
| dense (Dense) | (None, 32, 256) | 65792 |
| lstm_1 (LSTM) | (None, 32, 128) | 197120 |
| dense_1 (Dense) | (None, 32, 128) | 16512 |
| lstm_2 (LSTM) | (None, 32, 64) | 49408 |
| dense_2 (Dense) | (None, 32, 64) | 4160 |
| lstm_3 (LSTM) | (None, 32, 32) | 12416 |
| dense_3 (Dense) | (None, 32, 32) | 1056 |
| attention (attention) | (None, 32) | 64 |
| flatten (Flatten) | (None, 32) | 0 |
| dense_4 (Dense) | (None, 128) | 4224 |
| dense_5 (Dense) | (None, 256) | 33024 |
| dense_6 (Dense) | (None, 29) | 7453 |

Total params: 687,165
Trainable params: 687,165
Non-trainable params: 0

(a)

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| gru (GRU) | (None, 32, 256) | 222720 |
| dense (Dense) | (None, 32, 256) | 65792 |
| gru_1 (GRU) | (None, 32, 128) | 148224 |
| dense_1 (Dense) | (None, 32, 128) | 16512 |
| gru_2 (GRU) | (None, 32, 64) | 37248 |
| dense_2 (Dense) | (None, 32, 64) | 4160 |
| gru_3 (GRU) | (None, 32, 32) | 9408 |
| dense_3 (Dense) | (None, 32, 32) | 1056 |
| flatten (Flatten) | (None, 1024) | 0 |
| dense_4 (Dense) | (None, 256) | 262400 |
| dense_5 (Dense) | (None, 128) | 32896 |
| dense_6 (Dense) | (None, 64) | 8256 |
| dense_7 (Dense) | (None, 29) | 1885 |

Total params: 810,557
Trainable params: 810,557
Non-trainable params: 0

(b)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 32, 512)          591872
 l)

 dense (Dense)               (None, 32, 256)           131328

 bidirectional_1 (Bidirectio  (None, 32, 256)          394240
 nal)

 dense_1 (Dense)             (None, 32, 128)           32896

 bidirectional_2 (Bidirectio  (None, 32, 128)          98816
 nal)

 dense_2 (Dense)             (None, 32, 64)            8256

 bidirectional_3 (Bidirectio  (None, 32, 64)           24832
 nal)

 dense_3 (Dense)             (None, 32, 32)            2080

 flatten (Flatten)           (None, 1024)              0

 dense_4 (Dense)             (None, 128)               131200

 dense_5 (Dense)             (None, 256)               33024

 dense_6 (Dense)             (None, 29)                7453

=================================================================
Total params: 1,455,997
Trainable params: 1,455,997
Non-trainable params: 0
_________________________________________________________________
```

(c)

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional_4 (Bidirectio  (None, 32, 512)          445440
 nal)

 dense_7 (Dense)             (None, 32, 256)           131328

 bidirectional_5 (Bidirectio  (None, 32, 256)          296448
 nal)

 dense_8 (Dense)             (None, 32, 128)           32896

 bidirectional_6 (Bidirectio  (None, 32, 128)          74496
 nal)

 dense_9 (Dense)             (None, 32, 64)            8256

 bidirectional_7 (Bidirectio  (None, 32, 64)           18816
 nal)

 dense_10 (Dense)            (None, 32, 32)            2080

 flatten_1 (Flatten)         (None, 1024)              0

 dense_11 (Dense)            (None, 128)               131200

 dense_12 (Dense)            (None, 256)               33024

 dense_13 (Dense)            (None, 29)                7453

=================================================================
Total params: 1,181,437
Trainable params: 1,181,437
Non-trainable params: 0
_________________________________________________________________
```

(d)

**Figure 4.** (**a**) Description of LSTM implementation.; (**b**) Description of GRU implementation. (**c**) Description of Bi-LSTM implementation.; (**d**) Description of Bi-GRU implementation.

## 4. Experiments

### *4.1 Evaluation Phase*

The performance of classification models is evaluated and contrasted using these assessment measures. Precision, recall and F1-score provide information on particular elements, such as the trade-off between false positives and false negatives, while accuracy offers an overall assessment. When evaluating and applying these metrics, it's critical to take the problem's requirements and context into account.

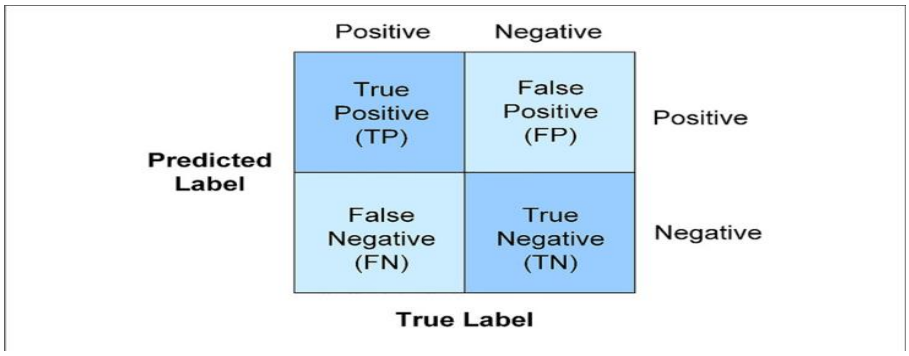



**Figure 5.** Confusion matrix.

**1- Accuracy:**

Accuracy is the ratio of accurately predicted instances to the total number of occurrences, and it indicates how accurate the model is overall in predicting future events. It offers a broad evaluation of the model's effectiveness in every class.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

**2- Precision:**

The precision of a model is measured by its ability to accurately identify positive instances out of all instances that are anticipated to be positive. It is helpful when there is a significant cost associated with false positives and centers on the accuracy of positive forecasts.

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

**3- Recall:**

The model's capacity to accurately identify positive cases out of all real positive instances is measured by the recall, which is sometimes referred to as sensitivity or true positive rate. It is useful when the cost of false negatives is substantial and concentrates on the completeness of positive predictions.

$$\textbf{Recall} = \frac{TP}{TP+FN}$$

**4- F1-score:**

This balanced indicator of the model's performance combines recall and precision into one number. It is the harmonic mean of recall and precision, placing equal weight on each measure.

$$\textbf{F1} - \textbf{score} = \frac{2(\text{Recall} * \text{Precision})}{\text{Recall} + \text{Precision}}$$

During the training process of deep learning, the loss value serves as an indicator of the model's performance. It quantifies the disparity between the model's predicted output and the actual target output. Computed through a loss function, the loss value measures the error by comparing predicted values against true values and deriving a single scalar value.

The primary objective throughout training is to minimize the loss value. By adjusting parameters like weights and biases, the model aims to reduce the loss and enhance its predictive capabilities. This optimization process frequently employs algorithms such as gradient descent, which iteratively updates the model's parameters based on the gradient of the loss function.

Monitoring the loss value is critical during training. As training advances, the loss generally diminishes, indicating that the model is learning and improving its performance. However, it's crucial to acknowledge that a low loss value during training does not guarantee favorable performance on unseen data. Overfitting is a common pitfall in deep learning, wherein the model becomes overly specialized to the training data and performs poorly on new data. To ensure the model's ability to generalize, it's imperative to evaluate its performance on separate validation or test sets.

### *4.2 Experimental Setup*

We conducted the studies in the Kaggle environment, modifying the settings so that NVIDIA TESLA P100 GPUs were used for all of the trials. While other procedures, like Pandas and scikit-learn libraries, may not benefit from GPU acceleration, these GPUs are particularly useful for training deep-learning models. A variety of Python libraries were also used, including TensorFlow for implementing and assessing the CNN model, CSV for reading Excel data files, Scikit-learn for using machine learning classifiers and producing evaluation metrics, and Kares for building and training the CNN model, among others.

### *4.3 Experimental Design*

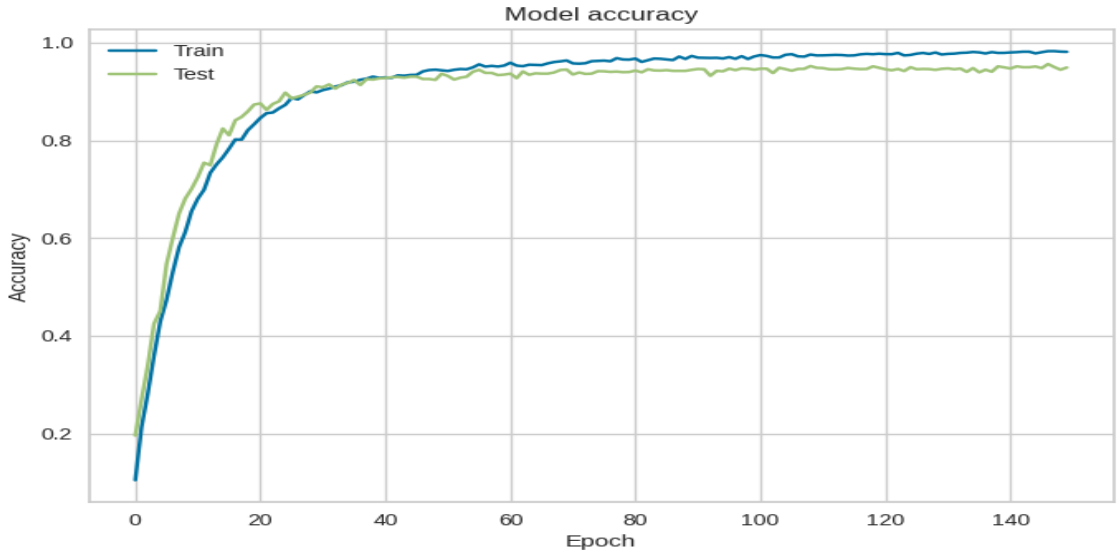
In this study, a machine learning model was constructed and trained using a deep learning framework. The model was compiled with a loss function of 'categorical_crossentropy' and an optimizer algorithm called 'Adam', while 'accuracy' was chosen as the metric to evaluate the model's performance. The training phase involved feeding the model with the provided training data, represented by the data set of the train and its label matrices. During training, a batch size of 128 was utilized, and the model was iterated over 150 epochs. Additionally, a validation dataset consisting of the data set of tests and its labels was used to assess the model's generalization ability. Throughout the training process, the model's weights were updated based on the specified loss function and optimizer, aiming to minimize the loss and improve prediction accuracy. By monitoring the validation data, overfitting was mitigated and the model's performance on unseen data was evaluated.

## 5. Results and Discussion

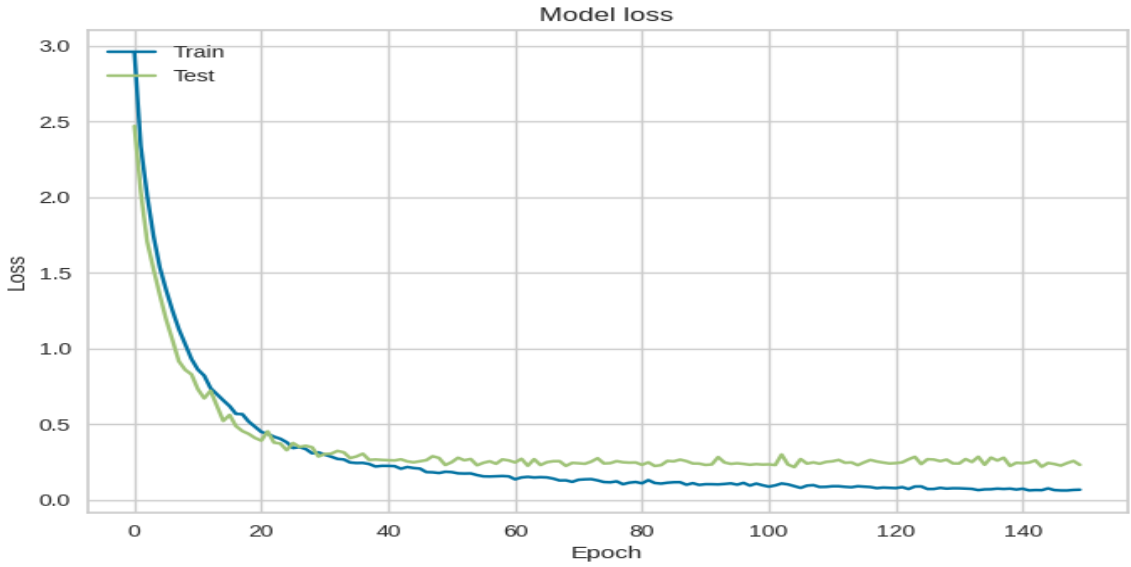
In this section, we present the outcomes achieved by the five aforementioned deep learning models on the previously mentioned datasets. These results serve to evaluate and compare the performance of the proposed models in recognizing handwritten Arabic characters. The purpose of Experiment 1 was to demonstrate the performance of the proposed models after they were trained and tested on the AHCD dataset. The outcomes of this experiment can be found in Table 2. Additionally, Figure 4 displays the accuracy, loss curves, and confusion matrix for both the training and testing phases of some proposed models.

**Table 2.** The results of proposed models on AHCD Dataset.

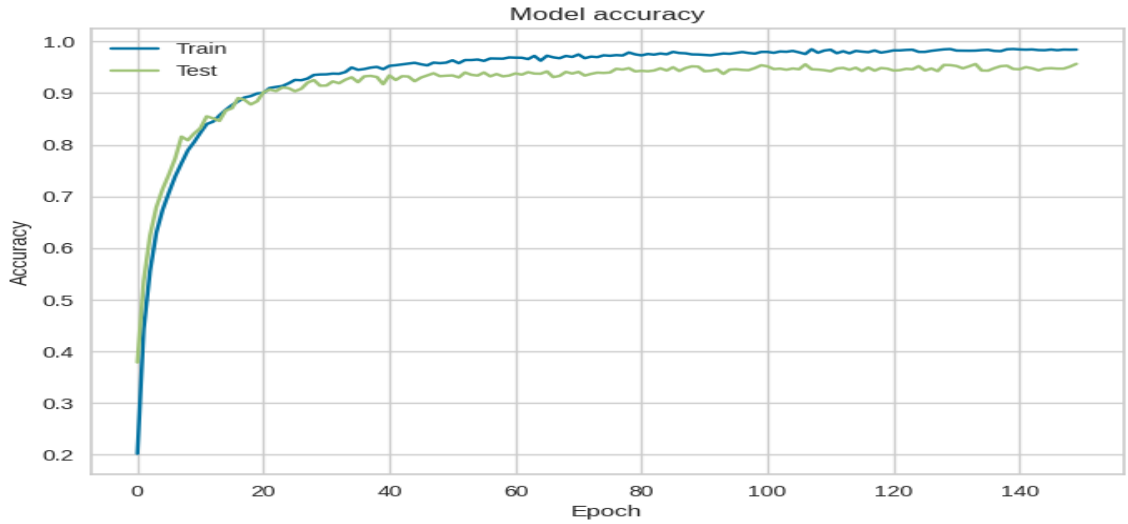
| Model | | CNN | | | LSTM | | | GRU | | | Bi-LSTM | | | Bi-GRU | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Character | Class # | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score |
| ا | 1 | 0.96 | 1 | 0.98 | 0.98 | 0.99 | 0.99 | 0.97 | 0.98 | 0.98 | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| ب | 2 | 0.95 | 0.98 | 0.97 | 1 | 0.96 | 0.98 | 0.94 | 0.98 | 0.96 | 0.99 | 0.97 | 0.98 | 0.98 | 0.97 | 0.97 |
| ت | 3 | 0.91 | 0.96 | 0.93 | 0.94 | 0.88 | 0.91 | 0.86 | 0.9 | 0.88 | 0.94 | 0.82 | 0.87 | 0.87 | 0.97 | 0.92 |
| ث | 4 | 0.94 | 0.94 | 0.94 | 0.9 | 0.97 | 0.93 | 0.91 | 0.89 | 0.9 | 0.88 | 0.95 | 0.92 | 0.94 | 0.94 | 0.94 |
| ج | 5 | 0.99 | 0.97 | 0.98 | 0.93 | 0.93 | 0.93 | 0.9 | 0.93 | 0.92 | 0.94 | 0.93 | 0.93 | 0.95 | 0.95 | 0.95 |
| ح | 6 | 0.89 | 0.98 | 0.94 | 0.91 | 0.92 | 0.91 | 0.85 | 0.93 | 0.89 | 0.93 | 0.93 | 0.93 | 0.9 | 0.97 | 0.93 |
| خ | 7 | 0.97 | 0.97 | 0.97 | 0.97 | 0.93 | 0.95 | 0.85 | 0.93 | 0.89 | 0.94 | 0.93 | 0.94 | 0.98 | 0.94 | 0.96 |
| د | 8 | 0.92 | 0.99 | 0.95 | 0.91 | 0.97 | 0.94 | 0.93 | 0.93 | 0.93 | 0.91 | 0.97 | 0.94 | 0.93 | 0.95 | 0.94 |
| ذ | 9 | 0.97 | 0.88 | 0.93 | 0.94 | 0.85 | 0.89 | 0.91 | 0.89 | 0.9 | 0.95 | 0.87 | 0.91 | 0.9 | 0.9 | 0.9 |
| ر | 10 | 0.94 | 0.98 | 0.96 | 0.92 | 0.96 | 0.94 | 0.91 | 0.95 | 0.93 | 0.92 | 0.95 | 0.93 | 0.94 | 0.97 | 0.96 |
| ز | 11 | 0.95 | 0.95 | 0.95 | 0.92 | 0.91 | 0.91 | 0.95 | 0.87 | 0.9 | 0.96 | 0.88 | 0.92 | 0.96 | 0.91 | 0.93 |
| س | 12 | 0.97 | 0.96 | 0.97 | 0.99 | 0.95 | 0.97 | 0.97 | 0.95 | 0.96 | 0.97 | 0.96 | 0.97 | 0.96 | 0.97 | 0.97 |
| ش | 13 | 0.99 | 0.94 | 0.97 | 0.99 | 0.97 | 0.98 | 0.96 | 1 | 0.98 | 0.96 | 1 | 0.98 | 0.98 | 0.97 | 0.97 |
| ص | 14 | 0.88 | 0.99 | 0.93 | 0.92 | 0.98 | 0.95 | 0.9 | 0.93 | 0.91 | 0.93 | 0.97 | 0.95 | 0.96 | 0.97 | 0.96 |
| ض | 15 | 0.99 | 0.9 | 0.94 | 0.96 | 0.94 | 0.95 | 0.97 | 0.93 | 0.94 | 0.98 | 0.94 | 0.96 | 0.98 | 0.92 | 0.95 |
| ط | 16 | 0.96 | 0.98 | 0.97 | 0.94 | 0.97 | 0.95 | 0.91 | 0.97 | 0.94 | 0.94 | 0.99 | 0.96 | 0.96 | 0.95 | 0.95 |
| ظ | 17 | 0.97 | 0.93 | 0.95 | 0.95 | 0.96 | 0.95 | 0.96 | 0.93 | 0.95 | 0.99 | 0.95 | 0.97 | 0.94 | 0.97 | 0.96 |
| ع | 18 | 0.96 | 0.9 | 0.93 | 0.9 | 0.93 | 0.92 | 0.95 | 0.83 | 0.89 | 0.93 | 0.97 | 0.95 | 0.97 | 0.96 | 0.96 |
| غ | 19 | 0.99 | 0.93 | 0.96 | 0.95 | 0.95 | 0.95 | 0.95 | 0.93 | 0.94 | 0.98 | 0.94 | 0.96 | 0.99 | 0.97 | 0.98 |
| ف | 20 | 0.9 | 0.98 | 0.94 | 0.93 | 0.96 | 0.94 | 0.91 | 0.92 | 0.91 | 0.85 | 0.97 | 0.91 | 0.93 | 0.97 | 0.95 |
| ق | 21 | 0.95 | 0.93 | 0.94 | 0.96 | 0.92 | 0.94 | 0.94 | 0.92 | 0.93 | 0.96 | 0.89 | 0.92 | 0.97 | 0.93 | 0.94 |
| ك | 22 | 0.97 | 0.97 | 0.97 | 0.99 | 0.98 | 0.99 | 0.99 | 0.95 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.98 | 0.98 |
| ل | 23 | 1 | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 0.96 | 0.97 | 0.97 | 0.98 | 0.99 | 0.98 | 0.99 | 0.98 | 0.99 |
| م | 24 | 0.96 | 0.97 | 0.96 | 0.95 | 0.99 | 0.97 | 0.97 | 0.97 | 0.97 | 0.98 | 1 | 0.99 | 0.99 | 1 | 1 |
| ن | 25 | 0.96 | 0.91 | 0.94 | 0.96 | 0.96 | 0.96 | 0.94 | 0.84 | 0.89 | 0.96 | 0.92 | 0.94 | 0.96 | 0.9 | 0.93 |
| ه | 26 | 0.94 | 0.98 | 0.96 | 0.95 | 0.95 | 0.95 | 0.95 | 0.96 | 0.95 | 0.96 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| و | 27 | 0.97 | 0.93 | 0.95 | 0.96 | 0.93 | 0.94 | 0.92 | 0.94 | 0.93 | 0.92 | 0.96 | 0.94 | 0.98 | 0.93 | 0.96 |
| ي | 28 | 1 | 0.95 | 0.97 | 0.97 | 0.97 | 0.97 | 0.96 | 0.92 | 0.94 | 0.98 | 0.97 | 0.98 | 0.96 | 0.96 | 0.96 |
| | | | | | | | | | | | | | | | | |
| **Accuracy (train)** | | | | 0.998 | | | 0.997 | | | 0.998 | | | 0.998 | | | **0.999** |
| **Accuracy (test)** | | | | 0.955 | | | 0.949 | | | 0.931 | | | 0.949 | | | **0.957** |
| **Loss (train)** | | | | 0.017 | | | 0.009 | | | 0.010 | | | 0.008 | | | **0.004** |
| **Loss (test)** | | | | 0.915 | | | **0.231** | | | 0.332 | | | 0.253 | | | 0.241 |
| Macro avg | | 0.96 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.93 | 0.93 | 0.93 | 0.95 | 0.95 | 0.95 | 0.96 | 0.96 | 0.96 |
| Weighted avg | | 0.96 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.93 | 0.93 | 0.93 | 0.95 | 0.95 | 0.95 | 0.96 | 0.96 | 0.96 |


Model accuracy
Train
Test
Accuracy
Epoch
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
120
140

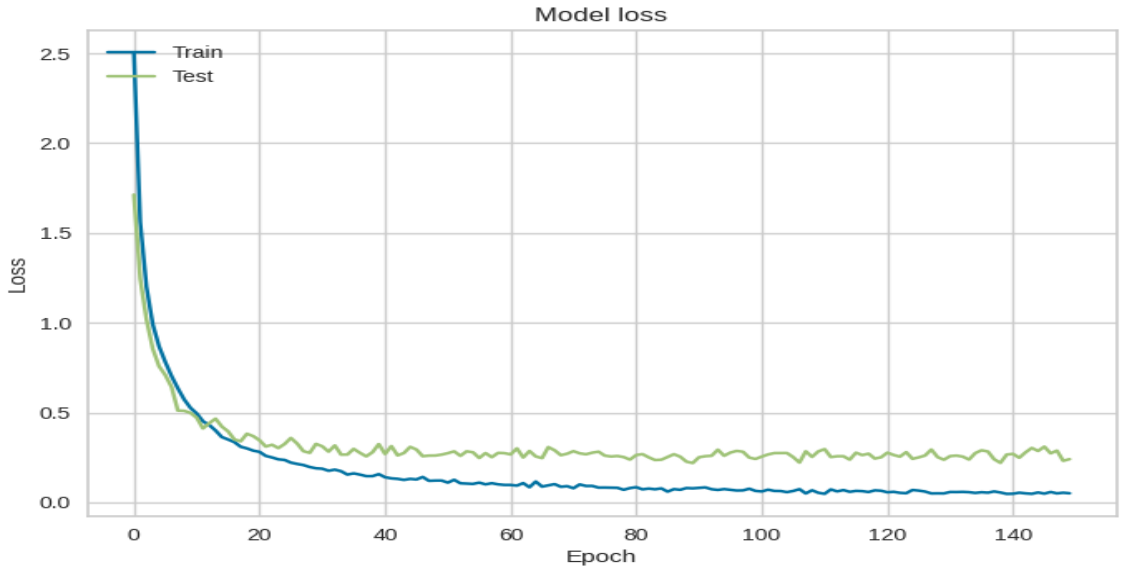
(**a**)


Model loss
Train
Test
Loss
Epoch
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100
120
140

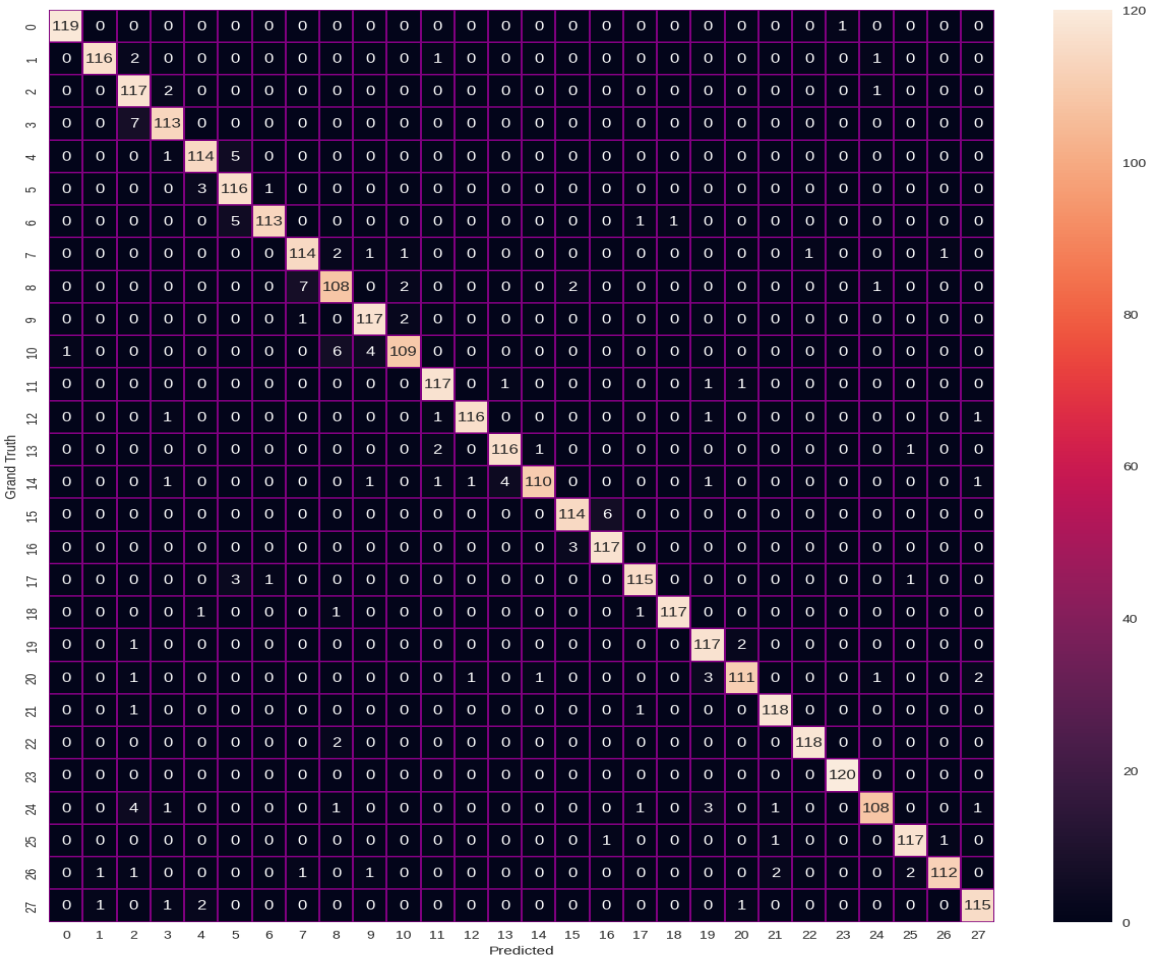
(**b**)


Model accuracy
Train
Test
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140

(c)




(d)



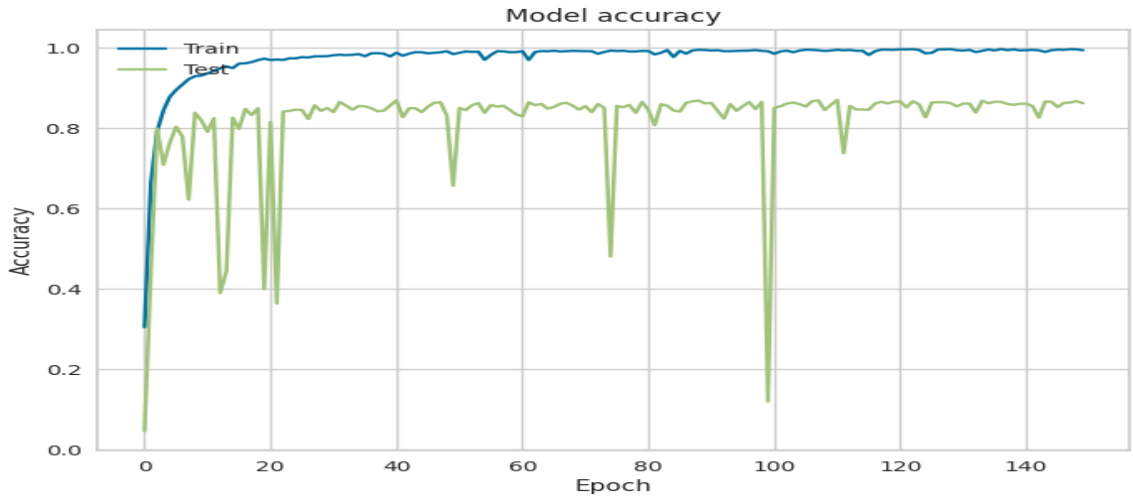

(e)

**Figure 4.** (**a**) Description of LSTM Model Accuracy curves of the training and testing.; (**b**) Description of LSTM Model Loss curves of the training and testing. (**c**) Description of Bi-GRU Model Accuracy curves of the training and testing.; (**d**) Description of Bi-GRU Model Loss curves of the training and testing; (**e**) Description of Bi-GRU Model confusion matrix.

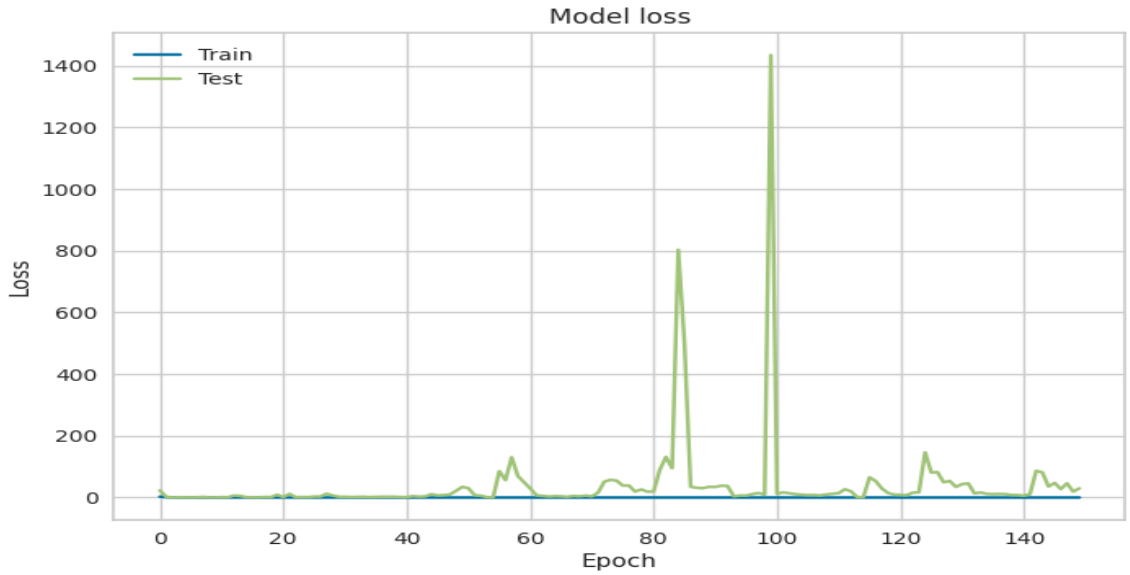
The purpose of Experiment 2 was to demonstrate the performance of the proposed models after their training and testing on the Hijjaa dataset. The results of this experiment are showcased in Table 3. Additionally, Figure 5 presents the accuracy, loss curves, and confusion matrix about the training and testing phases of some proposed models.

**Table 3.** The results of proposed Models on the Hijjaa dataset.

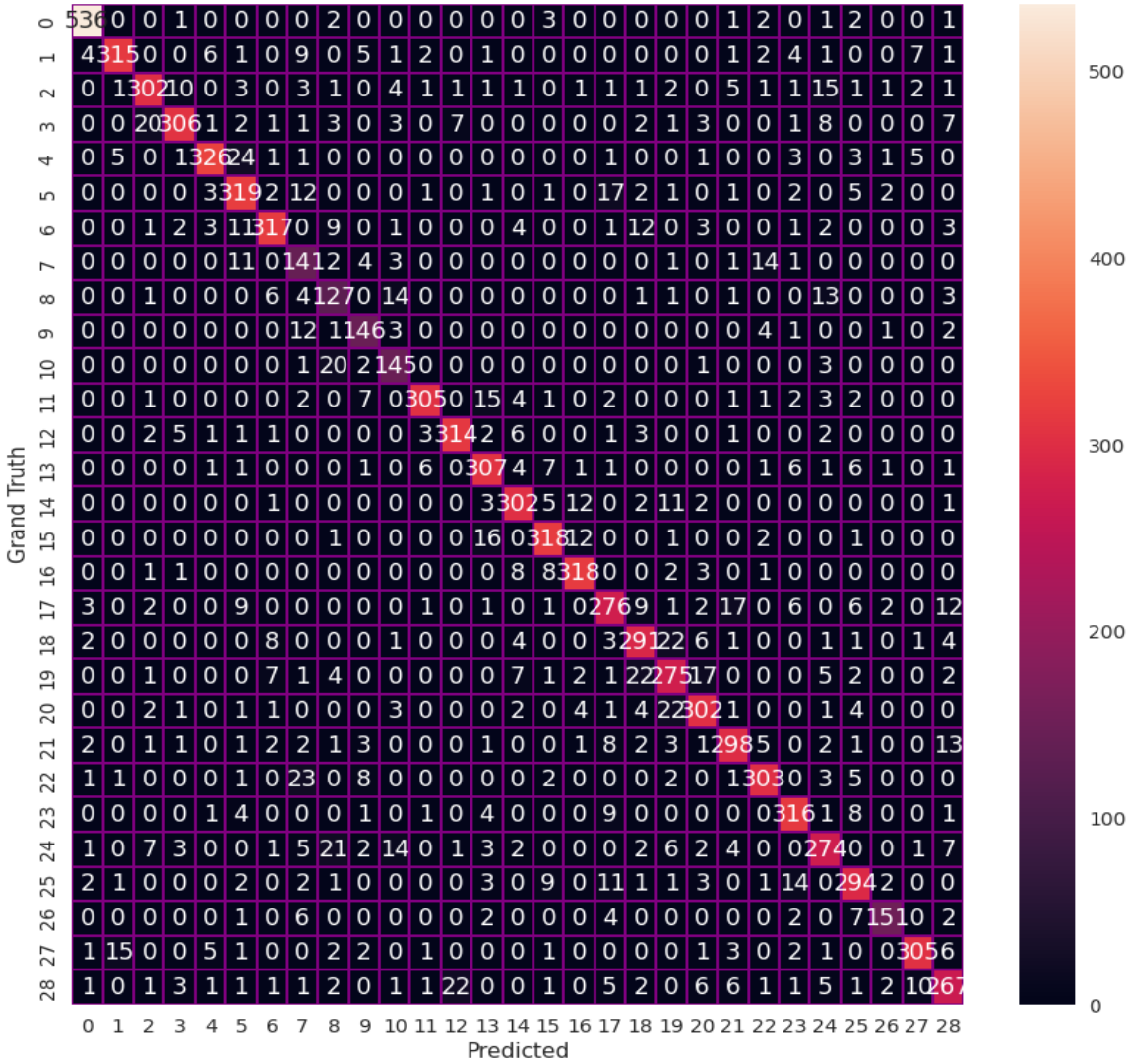
| Model | | CNN | | | LSTM | | | GRU | | | Bi-LSTM | | | Bi-GRU | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Character | Class # | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score |
| ا | 1 | 0.97 | 0.98 | 0.97 | 0.99 | 0.97 | 0.98 | 0.98 | 0.95 | 0.97 | 0.98 | 0.97 | 0.97 | 1 | 0.95 | 0.97 |
| ب | 2 | 0.93 | 0.88 | 0.9 | 0.95 | 0.93 | 0.94 | 0.95 | 0.89 | 0.92 | 0.95 | 0.93 | 0.94 | 0.95 | 0.92 | 0.93 |
| ت | 3 | 0.88 | 0.84 | 0.86 | 0.81 | 0.89 | 0.85 | 0.62 | 0.94 | 0.75 | 0.82 | 0.88 | 0.85 | 0.76 | 0.92 | 0.83 |
| ث | 4 | 0.92 | 0.84 | 0.87 | 0.84 | 0.86 | 0.85 | 0.78 | 0.75 | 0.77 | 0.87 | 0.84 | 0.85 | 0.87 | 0.83 | 0.85 |
| ج | 5 | 0.94 | 0.88 | 0.91 | 0.86 | 0.91 | 0.89 | 0.86 | 0.89 | 0.87 | 0.87 | 0.94 | 0.91 | 0.94 | 0.87 | 0.91 |
| ح | 6 | 0.81 | 0.86 | 0.84 | 0.72 | 0.83 | 0.77 | 0.82 | 0.73 | 0.77 | 0.78 | 0.81 | 0.79 | 0.81 | 0.8 | 0.81 |
| خ | 7 | 0.91 | 0.86 | 0.88 | 0.8 | 0.84 | 0.82 | 0.85 | 0.77 | 0.8 | 0.83 | 0.86 | 0.84 | 0.86 | 0.76 | 0.81 |
| د | 8 | 0.62 | 0.79 | 0.7 | 0.83 | 0.7 | 0.76 | 0.72 | 0.74 | 0.73 | 0.85 | 0.65 | 0.74 | 0.81 | 0.69 | 0.75 |
| ذ | 9 | 0.64 | 0.74 | 0.69 | 0.72 | 0.7 | 0.71 | 0.73 | 0.66 | 0.69 | 0.78 | 0.69 | 0.73 | 0.79 | 0.64 | 0.71 |
| ر | 10 | 0.81 | 0.86 | 0.83 | 0.91 | 0.85 | 0.88 | 0.94 | 0.78 | 0.85 | 0.88 | 0.89 | 0.89 | 0.9 | 0.88 | 0.89 |
| ز | 11 | 0.75 | 0.84 | 0.79 | 0.87 | 0.88 | 0.87 | 0.94 | 0.8 | 0.86 | 0.88 | 0.92 | 0.9 | 0.92 | 0.88 | 0.9 |
| س | 12 | 0.95 | 0.88 | 0.91 | 0.86 | 0.89 | 0.87 | 0.73 | 0.92 | 0.81 | 0.83 | 0.9 | 0.86 | 0.78 | 0.93 | 0.85 |
| ش | 13 | 0.91 | 0.92 | 0.91 | 0.91 | 0.87 | 0.89 | 0.9 | 0.86 | 0.88 | 0.9 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 |
| ص | 14 | 0.85 | 0.89 | 0.87 | 0.82 | 0.7 | 0.75 | 0.79 | 0.64 | 0.71 | 0.8 | 0.7 | 0.75 | 0.71 | 0.75 | 0.73 |
| ض | 15 | 0.88 | 0.89 | 0.88 | 0.86 | 0.79 | 0.82 | 0.85 | 0.7 | 0.77 | 0.7 | 0.87 | 0.78 | 0.77 | 0.83 | 0.8 |
| ط | 16 | 0.89 | 0.91 | 0.9 | 0.83 | 0.89 | 0.86 | 0.88 | 0.87 | 0.87 | 0.87 | 0.89 | 0.88 | 0.86 | 0.89 | 0.87 |
| ظ | 17 | 0.91 | 0.93 | 0.92 | 0.83 | 0.92 | 0.87 | 0.85 | 0.86 | 0.85 | 0.85 | 0.91 | 0.88 | 0.86 | 0.9 | 0.88 |
| ع | 18 | 0.81 | 0.79 | 0.8 | 0.81 | 0.69 | 0.75 | 0.75 | 0.72 | 0.73 | 0.84 | 0.72 | 0.77 | 0.81 | 0.73 | 0.77 |
| غ | 19 | 0.82 | 0.84 | 0.83 | 0.9 | 0.67 | 0.77 | 0.89 | 0.76 | 0.82 | 0.87 | 0.79 | 0.83 | 0.86 | 0.75 | 0.8 |
| ف | 20 | 0.78 | 0.79 | 0.79 | 0.71 | 0.78 | 0.75 | 0.72 | 0.74 | 0.73 | 0.8 | 0.7 | 0.74 | 0.72 | 0.82 | 0.77 |
| ق | 21 | 0.86 | 0.87 | 0.86 | 0.84 | 0.91 | 0.87 | 0.79 | 0.87 | 0.83 | 0.78 | 0.91 | 0.84 | 0.82 | 0.91 | 0.86 |
| ك | 22 | 0.87 | 0.86 | 0.86 | 0.84 | 0.89 | 0.86 | 0.8 | 0.86 | 0.83 | 0.85 | 0.89 | 0.87 | 0.84 | 0.89 | 0.86 |
| ل | 23 | 0.9 | 0.87 | 0.88 | 0.85 | 0.92 | 0.88 | 0.86 | 0.87 | 0.86 | 0.89 | 0.88 | 0.88 | 0.85 | 0.92 | 0.88 |
| م | 24 | 0.87 | 0.91 | 0.89 | 0.9 | 0.84 | 0.87 | 0.89 | 0.83 | 0.86 | 0.9 | 0.85 | 0.87 | 0.92 | 0.84 | 0.88 |
| ن | 25 | 0.8 | 0.77 | 0.79 | 0.78 | 0.74 | 0.76 | 0.65 | 0.73 | 0.69 | 0.8 | 0.71 | 0.75 | 0.79 | 0.73 | 0.76 |
| ه | 26 | 0.84 | 0.85 | 0.84 | 0.79 | 0.85 | 0.82 | 0.77 | 0.83 | 0.8 | 0.79 | 0.9 | 0.84 | 0.79 | 0.88 | 0.83 |
| و | 27 | 0.93 | 0.86 | 0.89 | 0.9 | 0.86 | 0.88 | 0.8 | 0.9 | 0.84 | 0.89 | 0.89 | 0.89 | 0.89 | 0.9 | 0.9 |
| ي | 28 | 0.92 | 0.88 | 0.9 | 0.93 | 0.95 | 0.94 | 0.9 | 0.9 | 0.9 | 0.95 | 0.92 | 0.93 | 0.92 | 0.94 | 0.93 |
| ء | 29 | 0.8 | 0.78 | 0.79 | 0.85 | 0.81 | 0.83 | 0.77 | 0.7 | 0.74 | 0.86 | 0.79 | 0.82 | 0.89 | 0.7 | 0.78 |
| | | | | | | | | | | | | | | | | |
| **Accuracy (train)** | | | | **0.995** | | | 0.962 | | | 0.927 | | | 0.983 | | | 0.973 |
| **Accuracy (test)** | | | | **0.863** | | | 0.846 | | | 0.815 | | | 0.850 | | | 0.846 |
| **Loss (train)** | | | | 1.351 | | | 0.121 | | | 0.258 | | | **0.052** | | | 0.082 |
| **Loss (test)** | | | | 29.765 | | | **0.609** | | | 0.670 | | | 0.689 | | | 0.662 |
| Macro avg | | 0.85 | 0.86 | 0.85 | 0.85 | 0.84 | 0.84 | 0.82 | 0.81 | 0.81 | 0.85 | 0.84 | 0.84 | 0.85 | 0.84 | 0.84 |
| Weighted avg | | 0.87 | 0.86 | 0.86 | 0.85 | 0.85 | 0.84 | 0.82 | 0.81 | 0.82 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 |



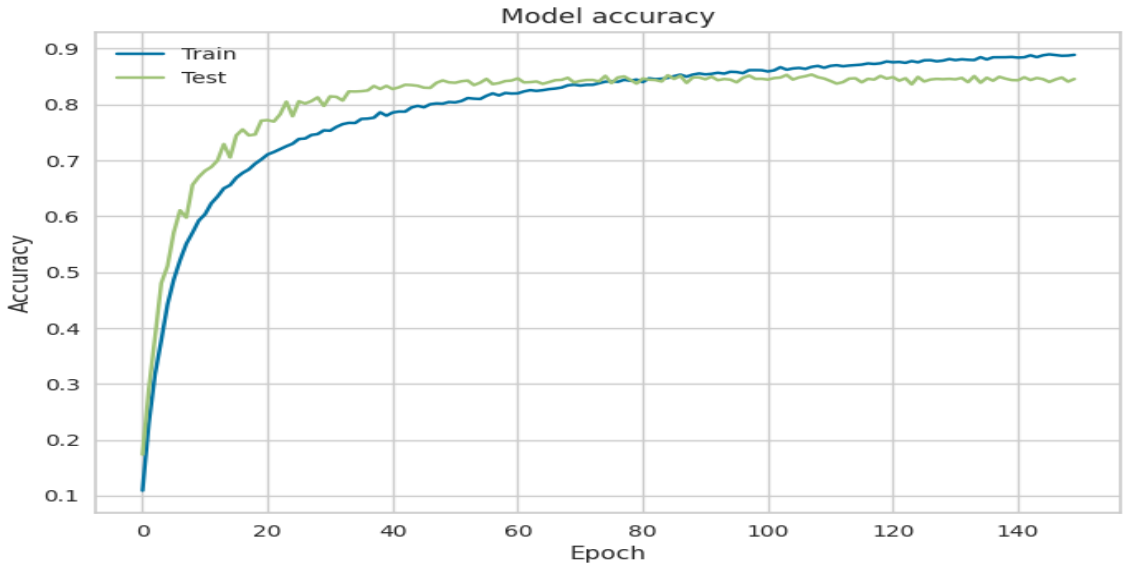

(a)


Model loss
Train
Test
Loss
Epoch
0
200
400
600
800
1000
1200
1400
0
20
40
60
80
100
120
140

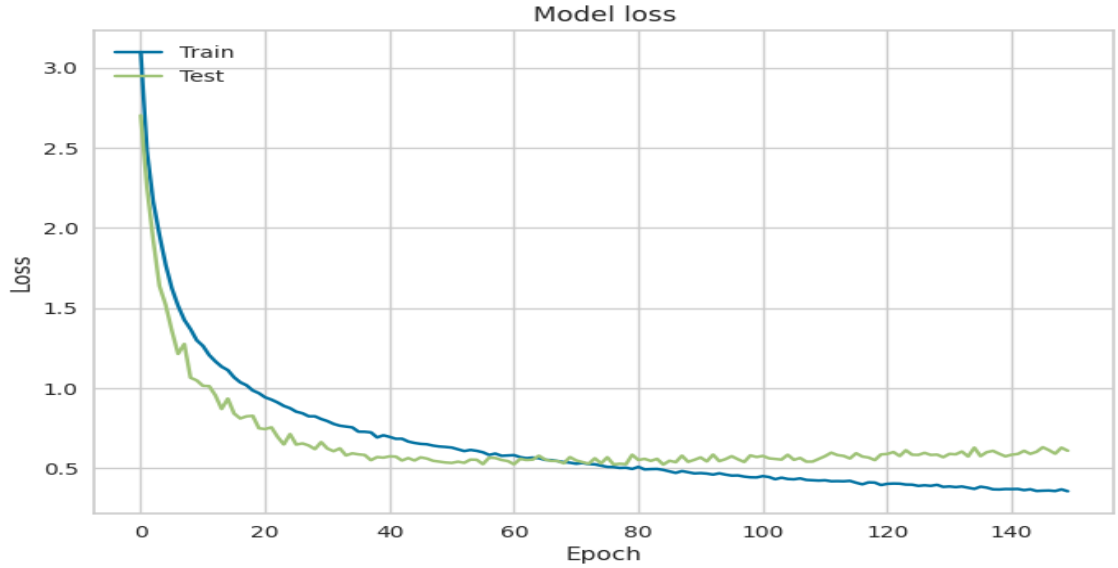
(b)


Grand Truth
Predicted
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
0
100
200
300
400
500

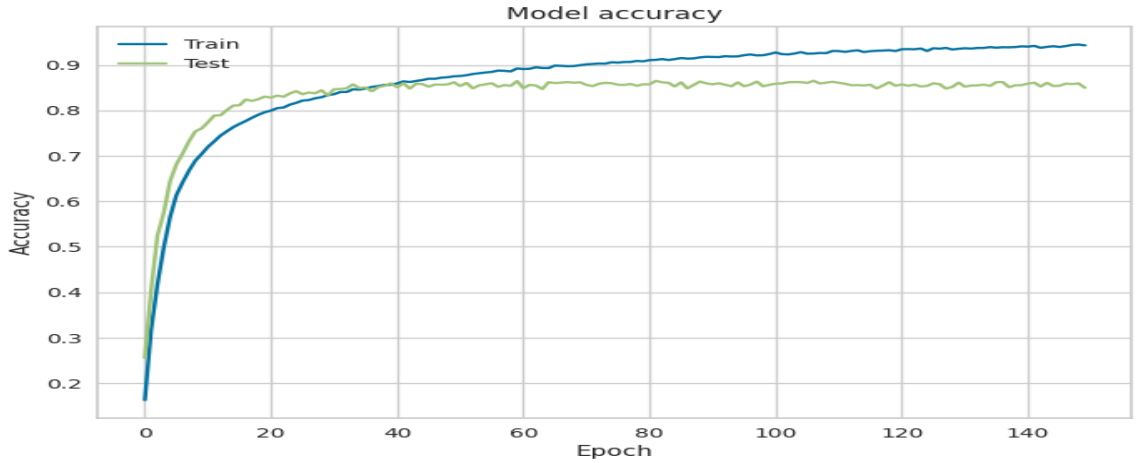
(c)


Model accuracy
Train
Test
Accuracy
Epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120
140

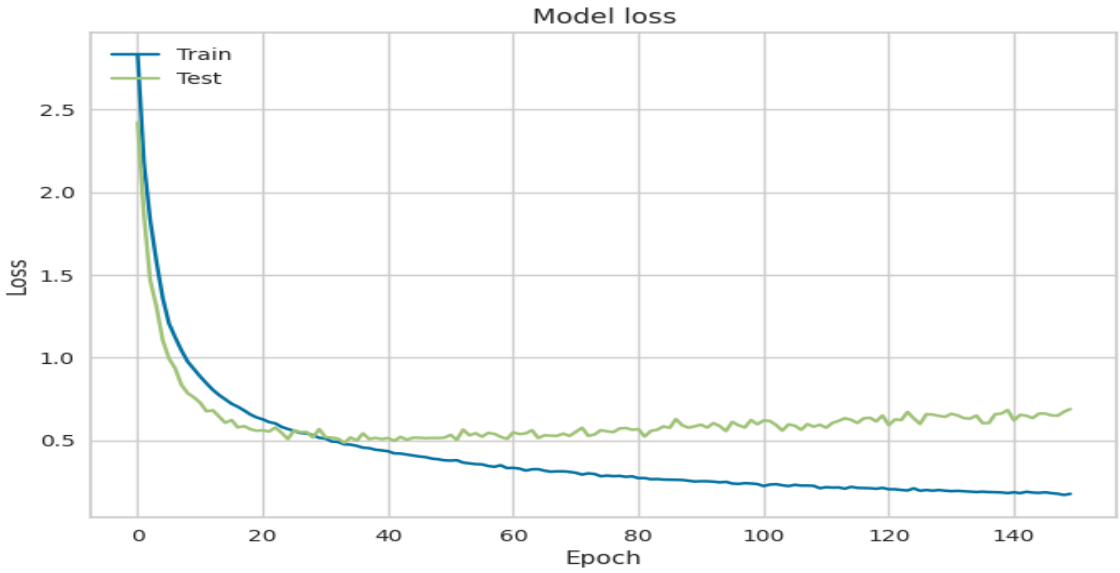
(d)


Model loss
Train
Test
Loss
Epoch
3.0
2.5
2.0
1.5
1.0
0.5
0
20
40
60
80
100
120
140

(e)


Model accuracy
Train
Test
Accuracy
Epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140

(f)




(g)

**Figure 5.** (**a**) Description of CNN Model Accuracy curves of the training and testing; (**b**) Description of CNN Model Loss curves of the training and testing. (**c**) Description of CNN Model confusion matrix.; (**d**) Description of LSTM Model Accuracy curves of the training and testing.; (**e**) Description of LSTM Model Loss curves of the training and testing.; (**f**) Description of Bi-LSTM Model Accuracy curves of the training and testing.; (**g**) Description of Bi-LSTM Model Loss curves of the training and testing.

Our research aimed to utilize existing deep-learning techniques and enhance the accuracy of Arabic handwriting recognition. To evaluate the performance of our proposed models, we utilized two datasets: AHCD, and Hijjaa. The evaluation process involved conducting two experiments. The first experiment involved classifying the AHCD dataset, which consists of 28 classes representing the letters in the Arabic alphabet. the second experiment aimed to classify the Hijjaa dataset, which includes 29 classes representing the letters in the alphabet, including hamza. All models were trained and tested on the AHCD, and Hijjaa datasets, with each model specifically designed for multiclass classification tasks. The models aimed to assign labels to the alphabet, with 28, and 29 labels for the AHCD, and Hijjaa datasets, respectively.

Our objective was to assess the performance of various Deep Learning models, namely CNN, LSTM, GRU, Bi-LSTM, and Bi-GRU, after training them with the AHCD, and Hijjaa datasets. Table 2 presents a comparison of their performance metrics specifically on the AHCD dataset. In terms of classification accuracy, precision, recall, and F1-score, Loss, the Bi-GRU outperformed all other models, achieving an accuracy of 95.7% on the testing set and 99.9% on the training set. Additionally, the Bi-GRU and LSTM models attained the lowest loss values of 0.231 and 0.004, respectively, on the test and training datasets. Table 2 also indicates that class numbers ranging from 0 to 27 correspond to the letters of the Arabic alphabet in sequence from "alif" to "yah". To visualize the accuracy curves, Figures 4(a), and 4(c) present the progress of LSTM, and Bi-GRU models, respectively, over the 150 epochs on the AHCD dataset. Similarly, Figures 4(b), and 4(d) depict the loss curves of LSTM, and Bi-GRU models, respectively, over the 150 epochs on the AHCD dataset. Finally, Figure 4(e) illustrates the Confusion Matrix of Bi-GRU, on the AHCD dataset.

Moving to Table 3, it compares the performance measures on the Hijjaa dataset. When considering classification accuracy, precision, recall, and F1-score, it was found that the CNN model surpassed all other models, achieving an accuracy of 86.3% on the testing set and 99.5% on the training set. Moreover, in terms of loss values, the Bi-LSTM and LSTM models achieved the lowest values on the training and testing datasets. The Bi-LSTM model attained a loss value of 0.052, while the GRU model achieved a loss value of 0.609. In Table 3, it can be observed that class numbers 0 to

27 are associated with the letters of the Arabic alphabet, starting with "alif" and ending with "yah." Additionally, class number 28 is assigned to represent "hamza". Figures 5(a), 5(d), and 5(f) depict the accuracy curves of CNN, LSTM, and Bi-LSTM models, respectively, over the 150 epochs on the Hijjaa dataset. Similarly, Figures 5(b), 5(e), and 5(g) illustrate the loss curves of CNN, LSTM, GRU, Bi-LSTM, and Bi-GRU models, respectively, over the 150 epochs on the Hijjaa dataset. Furthermore, Figure 5(c) demonstrates the Confusion Matrix of the CNN model, respectively, on the Hijjaa dataset.

**Table 4.** Performance comparison with models from the literature.

| References | Datasets | | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|
| **Altwaijry et al. [7]** | Hijjaa | | 88% | 88% | 88% | 88% |
| **Alrobah et al. [21]** | Hijjaa | | 96.3% | - | - | - |
| **Bin Durayhim et al. [23]** | Hijjaa | Train | **99.5%** | **99%** | **99%** | **99%** |
| | | Test | **86.3%** | **85%** | **86%** | **85%** |
| **Proposed LSTM** | Hijjaa | Train | 96.2% | 96% | 96% | 96% |
| | | Test | 84.6% | 85% | 84% | 84% |
| **Proposed GRU** | Hijjaa | Train | 92.7% | 92% | 92% | 92% |
| | | Test | 81.5% | 82% | 81% | 81% |
| **Proposed Bi-LSTM** | Hijjaa | Train | 98.3% | 98% | 98% | 98% |
| | | Test | 85% | 85% | 84% | 84% |
| **Proposed Bi-GRU** | Hijjaa | Train | 97.3% | 97% | 97% | 97% |
| | | Test | 84.6% | 85% | 85% | 85% |
| **Altwaijry et al. [7]** | AHCD | | 97% | 97% | 97% | 97% |
| **El-Sawy et al. [10]** | AHCD | | 94.9% | - | - | - |
| **Bin Durayhim et al. [23]** | AHCD | Train | 98% | 99% | 99% | 99% |
| | | Test | 95.5% | 95% | 95% | 95% |
| **Proposed LSTM** | AHCD | Train | 99.7% | 99% | 99% | 99% |
| | | Test | 94.9% | 95% | 95% | 95% |
| **Proposed GRU** | AHCD | Train | 99.8% | 99% | 99% | 99% |
| | | Test | 93.1% | 93% | 93% | 93% |
| **Proposed Bi-LSTM** | AHCD | Train | 99.8% | 99% | 99% | 99% |
| | | Test | 94.9% | 95% | 95% | 95% |
| **Proposed Bi-GRU** | AHCD | Train | **99.9%** | **99%** | **99%** | **99%** |
| | | Test | **95.7%** | **96%** | **96%** | **96%** |

In conclusion, we conducted a comparison of our classification results with those achieved by different models on various datasets as documented in the literature. Table 4 presents our results alongside the outcomes of other models, specifically on the Hijjaa and AHCD datasets.

In the study [23], the researcher relied on the output obtained from the training phase and used it as the value for testing. When applying the described model in this research, it was ensured that this conclusion was based on the training values, neglecting the importance of the test values, to prove that the obtained value is the best. After applying the research [23] and comparing it with other proposed models and the state of the art, the efficiency of the CNN model was proven compared to

other models in both the training and testing phases. The following models ranked in order: Bi-LSTM, LSTM, Bi-GRU, GRU when applied to the Hijjaa dataset. On the other hand, when applied to the AHCD dataset, the Bi-GRU model demonstrated efficiency compared to other models and the state of the art in both the training and testing phases, followed by CNN, LSTM, Bi-LSTM, and Bi-GRU.

## 6. Conclusions

In this research article, we introduced and evaluated five deep learning models: CNN, LSTM, GRU, Bi-LSTM, and Bi-GRU, specifically designed for Arabic handwriting character recognition. Our objective was to classify letters into 28, and 29 classes based on their shape. The evaluation was conducted using two datasets: AHCD, and Hijjaa, which contain Arabic handwriting samples. We compared the performances of the CNN, LSTM, GRU, Bi-LSTM, and Bi-GRU models among themselves, as well as against existing approaches in the literature.

Among the evaluated models, the CNN model emerged as the top performer, showcasing superior accuracy compared to state-of-the-art classification algorithms when trained and tested on the Hijjaa dataset. It achieved remarkable training accuracy of 99.5% and testing accuracy of 86.3%. Similarly, the Bi-GRU model stood out as the best performer when trained and tested on the AHCD dataset, achieving a training accuracy of 99.5% and a testing accuracy of 95.7%.

For future research, there are several avenues for expansion. It would be beneficial to extend the training of the developed deep learning models to encompass a combination of datasets, aiming to enhance overall performance. Furthermore, exploring additional supplementary features that may contribute to improving character recognition accuracy could prove valuable. Investigating intentional mistakes and analyzing their impact on the models' capabilities would also be an interesting direction. Additionally, applying these models in practical contexts, such as differentiating between children's and adults' handwriting or recognizing connected letters and words, could yield practical applications in writing, dictation, and calligraphy recognition.

**Funding**

This research was conducted without external funding support.

**Ethical approval**

This article does not contain any studies with human participants or animals performed by any of the authors.

**Conflicts of Interest**

The authors declare that there is no conflict of interest in the research

**.Institutional Review Board Statement**

Not applicable.

**Informed Consent Statement**

Not applicable.

**Data Availability Statement**

Not applicable

**Code Availability Statement**

The implementation used in this article was in GitHub. For details, please refer to "https://github.com/MohamedGresha/deep-learning-methods-on-Optical-Character-Recognition-Arabic-text/tree/main"

## References

[1] Ahmed, R., Dashtipour, K., Gogate, M., Raza, A., Zhang, R., Huang, K., Hawalah, A., Adeel, A., & Hussain, A. (2020). Offline arabic handwriting recognition using deep machine learning: A review of recent advances.

Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11691 LNAI, 457–468. https://doi.org/10.1007/978-3-030-39431-8_44

[2] Alghyaline, S. (2023). Arabic Optical Character Recognition: A Review. In CMES - Computer Modeling in Engineering and Sciences (Vol. 135, Issue 3, pp. 1825–1861). Tech Science Press. https://doi.org/10.32604/cmes.2022.024555

[3] Alrobah, N., & Albahli, S. (2022). Arabic Handwritten Recognition Using Deep Learning: A Survey. Arabian Journal for Science and Engineering, 47(8), 9943–9963. https://doi.org/10.1007/s13369-021-06363-3

[4] Mahdi, M. G., Sleem, A., & Elhenawy, I. (2024). Deep Learning Algorithms for Arabic Optical Character Recognition: A Survey. Multicriteria Algorithms with Applications, 2, 65–79. https://doi.org/10.61356/J.MAWA.2024.26861

[5] Albattah, W., & Albahli, S. (2022). Intelligent Arabic Handwriting Recognition Using Different Standalone and Hybrid CNN Architectures. Applied Sciences 2022, Vol. 12, Page 10155, 12(19), 10155. https://doi.org/10.3390/APP121910155

[6] Ali, A. A. A., Suresha, M., & Ahmed, H. A. M. (2020). A Survey on Arabic Handwritten Character Recognition. SN Computer Science, 1(3), 1–10. https://doi.org/10.1007/S42979-020-00168-1/TABLES/4

[7] Altwaijry, N., & Al-Turaiki, I. (2021). Arabic handwriting recognition system using convolutional neural network. Neural Computing and Applications, 33(7), 2249–2261. https://doi.org/10.1007/s00521-020-05070-8

[8] Balaha, H. M., Ali, H. A., & Badawy, M. (2021). Automatic recognition of handwritten Arabic characters: a comprehensive review. Neural Computing and Applications, 33(7), 3011–3034. https://doi.org/10.1007/S00521-020-05137-6/TABLES/13

[9] Ghanim, T. M., Khalil, M. I., & Abbas, H. M. (2020). Comparative Study on Deep Convolution Neural Networks DCNN-Based Offline Arabic Handwriting Recognition. IEEE Access, 8, 95465–95482. https://doi.org/10.1109/ACCESS.2020.2994290

[10] El-Sawy, A., Loey, M., & El-Bakry, H. (2017). Arabic handwritten characters recognition using convolutional neural network. WSEAS Transactions on Computer Research, 5(1), 11–19.

[11] Balaha, H. M., Ali, H. A., Saraya, M., & Badawy, M. (2021). A new Arabic handwritten character recognition deep learning system (AHCR-DLS). Neural Computing and Applications, 33(11), 6325–6367. https://doi.org/10.1007/S00521-020-05397-2/FIGURES/33

[12] de Sousa, I. P. (2018). Convolutional ensembles for Arabic Handwritten Character and Digit Recognition. PeerJ Computer Science, 2018(10), e167. https://doi.org/10.7717/PEERJ-CS.167/SUPP-4

[13] Boufenar, C., Kerboua, A., & Batouche, M. (2018). Investigation on deep learning for off-line handwritten Arabic character recognition. Cognitive Systems Research, 50, 180–195. https://doi.org/10.1016/J.COGSYS.2017.11.002

[14] Ullah, Z., & Jamjoom, M. (2022). An intelligent approach for Arabic handwritten letter recognition using convolutional neural network. PeerJ Computer Science, 8, e995. https://doi.org/10.7717/PEERJ-CS.995/SUPP-2

[15] Alyahya, H., Ismail, M. M. Ben, & Al-Salman, A. (2020). Deep ensemble neural networks for recognizing isolated Arabic handwritten characters. ACCENTS Transactions on Image Processing and Computer Vision, 6(21), 68–79. https://doi.org/10.19101/TIPCV.2020.618051

[16] Aljarrah, M. N., Zyout, M. M., & Duwairi, R. (2021). Arabic Handwritten Characters Recognition Using Convolutional Neural Network. 2021 12th International Conference on Information and Communication Systems, ICICS 2021, 182–188. https://doi.org/10.1109/ICICS52457.2021.9464596

[17] Ali, A. A. A., & Mallaiah, S. (2022). Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout. Journal of King Saud University - Computer and Information Sciences, 34(6), 3294–3300. https://doi.org/10.1016/J.JKSUCI.2021.01.012

[18] Alkhateeb, J. H. (2020). An effective deep learning approach for improving off-line arabic handwritten character recognition. International Journal of Software Engineering and Computer Systems, 6(2), 53–61.

[19] Nayef, B. H., Abdullah, S. N. H. S., Sulaiman, R., & Alyasseri, Z. A. A. (2022). Optimized leaky ReLU for handwritten Arabic character recognition using convolution neural networks. Multimedia Tools and Applications, 81(2), 2065–2094. https://doi.org/10.1007/s11042-021-11593-6

[20] Wagaa, N., Kallel, H., & Mellouli, N. (2022). Improved Arabic Alphabet Characters Classification Using Convolutional Neural Networks (CNN). Computational Intelligence and Neuroscience, 2022, 9965426. https://doi.org/10.1155/2022/9965426

[21] Alrobah, N., & Albahli, S. (2021). A Hybrid Deep Model for Recognizing Arabic Handwritten Characters. IEEE Access, 9, 87058–87069. https://doi.org/10.1109/ACCESS.2021.3087647

[22] Bouchriha, L., Zrigui, A., Mansouri, S., Berchech, S., & Omrani, S. (2022). Arabic Handwritten Character Recognition Based on Convolution Neural Networks. Communications in Computer and Information Science, 1653 CCIS, 286–293. https://doi.org/10.1007/978-3-031-16210-7_23

[23] Bin Durayhim, A., Al-Ajlan, A., Al-Turaiki, I., & Altwaijry, N. (2023). Towards Accurate Children's Arabic Handwriting Recognition via Deep Learning. Applied Sciences 2023, Vol. 13, Page 1692, 13(3), 1692. https://doi.org/10.3390/APP13031692

[24] Younis, K. S. (2017). Arabic hand-written character recognition based on deep convolutional neural networks. Jordanian Journal of Computers and Information Technology, 3(3).

[25] AlJarrah, M. N., Mo'ath, M. Z., & Duwairi, R. (2021). Arabic handwritten characters recognition using convolutional neural network. 2021 12th International Conference on Information and Communication Systems (ICICS), 182–188.

**Received:** 01 Mar 2024, **Revised:** 17 Apr 2024,

**Accepted:** 12 May 2024, **Available online:** 13 Jun 2024.

© 2024 by the authors. Submitted for possible open-access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

**Disclaimer/Publisher's Note:** The perspectives, opinions, and data shared in all publications are the sole responsibility of the individual authors and contributors, and do not necessarily reflect the views of Sciences Force or the editorial team. Sciences Force and the editorial team disclaim any liability for potential harm to individuals or property resulting from the ideas, methods, instructions, or products referenced in the content.