PeerJ Computer Science

An intelligent approach for Arabic handwritten letter recognition using convolutional neural network

Zahid Ullah[1] and Mona Jamjoom[2]

[1] Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

[2] Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia

ABSTRACT

Currently, digital transformation has occurred in most countries in the world to varying degrees, but digitizing business processes are complex in terms of understanding the various aspects of manual documentation. The use of digital devices and intelligent systems is vital in the digital transformation of manual documentation from hardcopy to digital formats. The transformation of handwritten documents into electronic files is one of the principal aspects of digitization and represents a common need shared by today's businesses. Generally, handwriting recognition poses a complex digitization challenge, and Arabic handwriting recognition, specifically, proves inordinately challenging due to the nature of Arabic scripts and the excessive diversity in human handwriting. This study presents an intelligent approach for recognizing handwritten Arabic letters. In this approach, a convolution neural network (CNN) model is proposed to recognize handwritten Arabic letters. The model is regularized using batch normalization and dropout operations. Moreover, the model was tested with and without dropout, resulting in a significant difference in the performance. Hence, the model overfitting has been prevented using dropout regularization. The proposed model was applied to the prominent, publicly-available Arabic handwritten characters (AHCD) dataset with 16,800 letters, and the performance was measured using several evaluation measures. The experimental results show the best fit of the proposed model in terms of higher accuracy results that reached 96.78%; additionally, other evaluation measures compared to popular domain-relevant approaches in the literature.

Subjects Algorithms and Analysis of Algorithms, Artificial Intelligence, Computer Vision, Data Mining and Machine Learning, Natural Language and Speech

Keywords Arabic handwritten letter, Recognition, Intelligent model, CNN, Evaluation

Submitted 28 March 2022
Accepted 6 May 2022
Published 27 May 2022

Corresponding author
Mona Jamjoom,
mmjamjoom@pnu.edu.sa

Academic editor
Muhammad Asif

Additional Information and Declarations can be found on page 18

DOI **10.7717/peerj-cs.995**

INTRODUCTION

The world today is embracing digitizing documents through the adoption of various automated handwritten document recognition techniques (*Balaha et al., 2021*). In addition, an automated handwritten recognition system takes part in many ancillary applications, such as: educational applications (*Altwaijry & Al-Turaiki, 2021*), friendly learning environment (*Al-Helali & Mahmoud, 2017*), bank cheque handling, reading application forms, postal address handling, and handwriting-to-speech transformation

(*Ahmed et al., 2020*). Consequently, handwriting recognition remains an active research area in recent decades and lots of handwriting recognition systems have been introduced to recognize different languages, of which the most common were English (*Yuan et al., 2012*), Chinese (*Xiao et al., 2017*; *Zhong, Jin & Feng, 2015*; *Bai et al., 2014*), French (*Xiao et al., 2020*), Urdu (*Ali et al., 2020*), and Arabic (*Younis, 2017*; *Shams, Elsonbaty & El Sawy, 2020*; *Balaha et al., 2021*). Furthermore, different research studies have proposed various intelligent techniques for handwriting recognition using machine learning methods, wherein some research has focused on digits, characters, text, or these elements in combination (*Mudhsh & Almodfer, 2017*). Moreover, some studies have applied their systems offline, such as in *Ahmed et al. (2021)*, *Yuan et al. (2012)* and *Abdalkafor & AlHamouz (2016)* while others apply them online (*Al-Taani & Al-Haj, 2010*; *Al-Helali & Mahmoud, 2017*).

Currently, Arabic is one of the most common international languages used in the Middle East and North Africa (*Javed, 2013*), and one of the most widespread spoken languages in the world; its speakers exceed half a billion over the world (*Altwaijry & Al-Turaiki, 2021*). Moreover, many languages use Arabic letters, words, and sentence structures, such as Urdu, Pashto, Persian, and Jawi (*Younis, 2017*; *Balaha, Ali & Badawy, 2021*). The standard written Arabic language consists of 28 letters which are usually written starting from right to left and were derived from the script of the holy Quran (*Javed, 2013*; *Balaha, Ali & Badawy, 2021*). Some letters may look different if they occur at the start, middle, or end of the word; that is, the shape of the letter is changed based on its location in the word (*Al-Taani & Al-Haj, 2010*).

A general problem facing any handwriting recognition language system is the variability of the font and the letter dimension across the handwriting from one individual to another (*Sahloul & Suen, 2014*). Some studies have reported during data collection that numerous variabilities can be observed in the handwriting of the same individual (*Almansari & Hashim, 2019*), thus automatic recognition of Arabic letters remains a challenge (*Mustapha et al., 2022*). Likewise, Arabic handwriting recognition poses many challenges because of: (1) high diversity in human handwriting, (2) shortage of large and public datasets, (3) the nature of Arabic script (*e.g.*, cursive form, variable letter size, different letter's shape depends on its location…etc.) (*El-Sawy, Loey & EL-Bakry, 2017*; *Al-Taani & Al-Haj, 2010*), (4) diverse styles in handwritten (*Al-Helali & Mahmoud, 2017*) and (5) There is a wealth of handwritten Arabic documents available in libraries (*Balaha et al., 2021*).

This study presents an intelligent approach to handwritten Arabic letter recognition using CNN. The proposed model developed in this approach was applied to a publicly available prominent dataset, referred to as AHCD. The proposed model has to recognize the 28 Arabic letters that exist in the dataset. The experimental results show the high performance of the proposed model applied to the AHCD dataset.

The remainder of this paper is ordered as follows: the next section discusses the literature review followed by step-by-step methodology, the next is the experimental setup followed by results and discussion, and the last section concludes this study.

LITERATURE REVIEW

In this section, we summarized the works in the literature which introduced different methods for Arabic handwritten letter recognition. We focused on single letter recognition since it's our concern in this study. *Sahloul & Suen (2014)* believed that achieving good accuracy was mainly based on the features selection during the preprocessing phase in the handwriting character recognition process. For that, they have used the CENPARMI database with 11,620 letters and extracted statistical, morphological and topological features. The method has shown a recognition rate of about 88% for all letters. In *Abdalkafor & AlHamouz (2016)*, used the same dataset with a novel feature selection approach and MLP neural network and reported the model accuracy of 94.75%.

Similarly, *Elleuch, Maalej & Kherallah (2016)* have proposed an integrated model of two classifiers: CNN and Support vector machine (SVM) and used dropout for securing the model from overfitting. They found that CNN-based SVM with dropout is more efficient in recognizing the Arabic letters than that without dropout. Furthermore, they have tested the model on Handwritten Arabic characters (HACDB) and IFN/ENIT datasets with dropout operation and found error classification rates of 5.83% and 7.05% respectively.

In *El-Sawy, Loey & EL-Bakry (2017)* the authors have developed a CNN model for Arabic letter recognition. The authors have utilized multiple optimization methods for enhancing the performance of the model. The model was employed on the AHCD dataset and reported a testing accuracy of 94.9%. *Younis (2017)* introduced a deep neural network model using a robust CNN for handwritten Arabic character recognition. The proposed CNN used the regularization parameter for overfitting avoidance. The proposed model was applied to AIA9K and AHCD datasets and reported the classification error for both datasets of 5.2% and 2.4%, respectively.

Boufenar, Kerboua & Batouche (2018) investigated the applicability of the proposed DCNN for the recognition of offline handwritten Arabic letters using a transfer learning strategy. They have tested the proposed model on OIHACDB and AHCD datasets. The authors of this study have enriched the AHCD datasets with additional samples in the training set and reported a high level of efficacy of the proposed method. *Elleuch & Kherallah (2018)* has presented a deep learning based on SVM (DeepSVM) model for recognizing handwritten Arabic letters. In this regard, multi-label SVM and RBF kernel were used for testing Arabic letters in the HACDB dataset. The authors concluded with the effectiveness of the proposed method in terms of the experimental results and reported an error classification rate of 8.64%. *Almansari & Hashim (2019)* have presented a deep learning architecture with CNN and MLP neural network for recognizing isolated handwritten Arabic letters. The architecture performance was evaluated using the AHCD dataset and concluded with a testing accuracy of 95.27% of the architecture.

Another attempt was made by *Ali & Suresha, (2019)* for recognizing the Arabic handwritten letters using machine learning methods. An integrated recognition system based on CNN, k-Nearest Neighbour (kNN), and SVM was developed, and the accuracies analysis and classifiers' performance were measured. The integrated recognition system

was applied to AHDB and IFN/ENIT databases and reported an accuracy of 98.4%. This is worth noting that the datasets used were different from our study.

In *Shams, Elsonbaty & El Sawy (2020)*, the authors proposed a method to recognize, detect, and validate input handwritten Arabic characters using deep CNN and SVM. The proposed method attempted to check the similarity between the input characters and the stored characters. It used a clustering method specifically k-means for recognizing multi-stroke characters and applied the proposed method on 840 tested images. The reported result of model accuracy was 95.07%. *Djaghbellou et al. (2020)* has proposed a new framework based on multi-scale histogram oriented gradient (HOG) features and the deep rule-based classifier (DRB) for recognizing handwritten Arabic letters. The framework integrates multi-scale HOG during feature extraction and then DRB is used to classify the comprehensive HOG features and predict the class labels. The framework was applied to the AHCD dataset and reported the recognition rate of 74.61% which was comparatively higher than other studies.

A larger dataset named "HMBD" that contains 54,115 characters has been used by *Balaha et al. (2021)*. After multiple attempts, they selected the best 14 native CNN architectures that differ in a hierarchy of constructing the architecture layers, the best of them resulted in a testing accuracy of 91.96%. Furthermore, more improvement has been done using optimization methods (*e.g.*, transfer learning (TF) and genetic algorithm (GA)) that led to achieving a testing accuracy of 92.88%. *Ali & Mallaiah (2021)* have proposed a model of CNN based-SVM with dropout for Arabic handwritten letter recognition and tested on several datasets such as AHDB, AHCD, HACDB, and IFN/ENIT. The authors reported promising results of the proposed model in comparison to other models built for the same domain. The model achieved an accuracy of 96.80% applied on AHDB.

A novel supervised CNN model was introduced by *Ahmed et al. (2021)* that extracts optimal features and applied batch normalization and dropout regularization layers for enhancing the overall performance. The model has shown excellent classification results on a diverse set of six benchmark databases, including MADBase (Digits), CMATERDB (Digits), HACDB (Characters), SUST-ALT (Digits), SUST-ALT (Characters), and SUST-ALT (Names). Although, the accuracy of this model is above 99% but the model was not tested on the AHCD dataset.

An attempt was made by *Aljarrah, Zyout & Duwairi (2021)* who proposed a CNN model for recognizing the Arabic offline letters. The proposed model was trained using the AHCD dataset and achieved high performance model with an error classification rate of 2.3% after applying data augmentation. Subsequently, the authors (*Altwaijry & Al-Turaiki, 2021*) evaluated their offline model for recognizing Arabic letters based on CNN architecture using two datasets: Hijja which contains 47,434 images of single Arabic characters written by children aged 7–12 years and AHCD containing 16,800 images. The performance of the model shows an error classification rate of 3% and 12% for AHCD and Hijja, respectively.

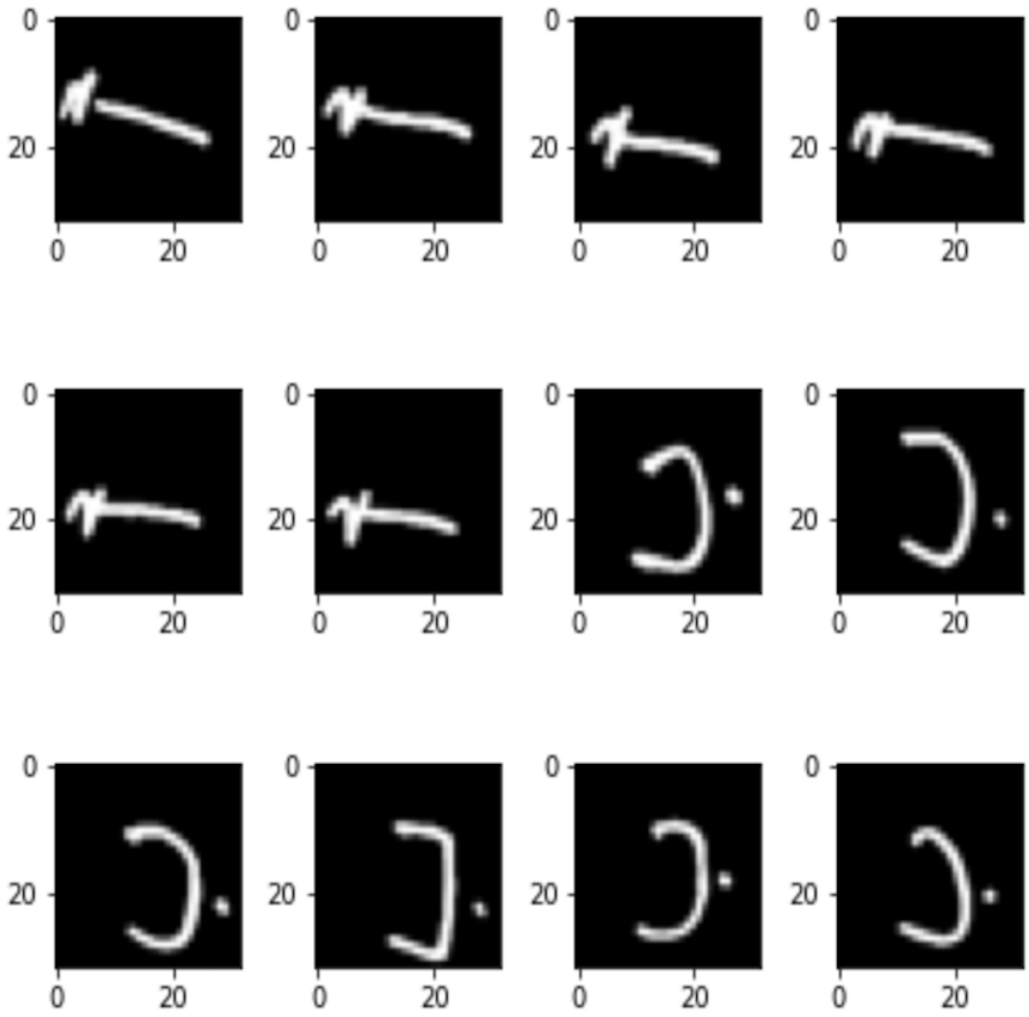

Figure 1 **Sample of Arabic letters in a training set.**
Full-size DOI: 10.7717/peerjcs.995/fig-1

MATERIALS & METHODS

Data collection

The dataset used to conduct this study was the Arabic handwritten characters dataset (AHCD) harvested from Kaggle (*El-Sawy, Loey & EL-Bakry, 2017*). The dataset consisted of a total of 16,800 handwritten letters collected from 60 people ranging in age from 19 to 40 years. The dataset comprised 4 separate files (*i.e.,* train images, training labels, test images, and testing labels). The authors have further explained that 90% of the people were right-handed. The total number of Arabic class labels used in AHCD is 28 (*i.e.,* from 'alef' to 'yaa'). During the data collection, each person was required to provide a set of 28 letters repeatedly 10 times. Figure 1 shows sample letters in the AHCD dataset. Moreover, the 16,800 letters were split into two sets in that 80% of the letters were in training and the rest of 20% were in the test set. The training set consisted of 13,440 letters divided into 480

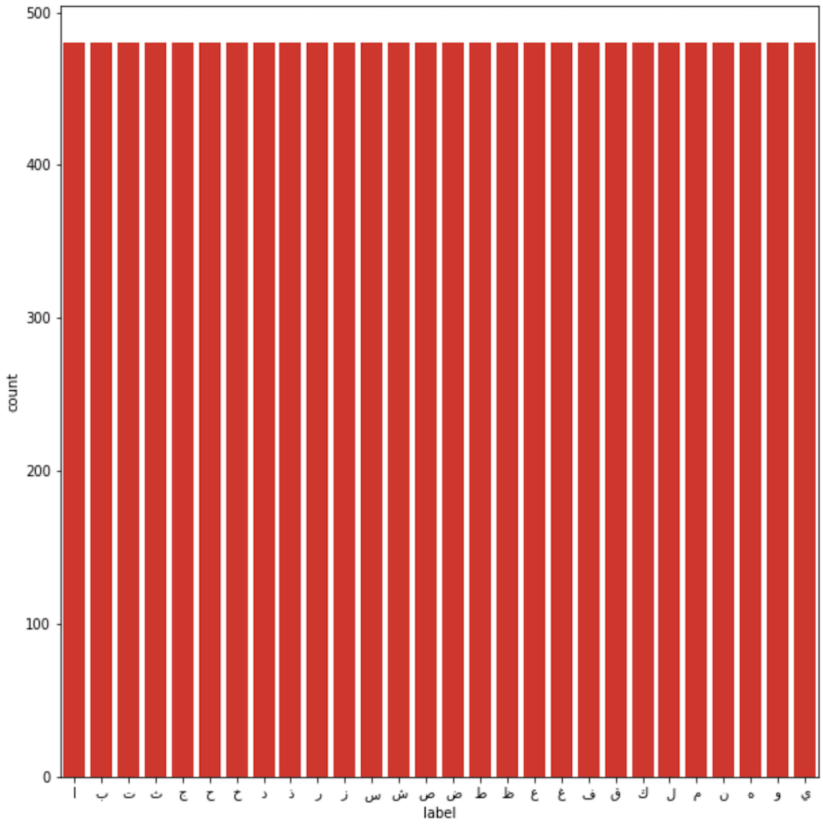

Figure 2 **Training images per class.**

Full-size DOI: 10.7717/peerjcs.995/fig-2

images per class and the test set of 3,360 letters divided into 120 images per class as shown in Figs. 2 and 3 respectively.

Dataset preprocessing

Data preprocessing is an essential step in enhancing the quality of data by removing "garbage" (*Al-Mudimigh & Ullah, 2011*) and preparing it for best-fitted model development (*Ullah & Al-Mudimigh, 2012*). In this study, firstly, the images were reshaped in size to 32× 32. According to *Ghosh, Das & Nasipuri (2019)*, CNN is normally accepting fixed size images that leads to several challenges in data acquisition and model development; therefore, the input image reshape is performed to overcome such challenges and fed to the network. Similarly, flipping and rotations were performed for better performance and then reshaped the whole array to a size of 32× 32× 1. The 32× 32× 1 size image represents a greyscale image. Moreover, the images were also normalized (or simply divided by 255). The categorical class labels were also executed in that integer values that represent different categories were transformed into a matrix of binary values (*Ullah & Jamjoom, 2022*). For a better understanding of the classifier, the index of the class labels should start at 0 rather than 1 in the original dataset; therefore, the start of the class label is set to 0, thus the label

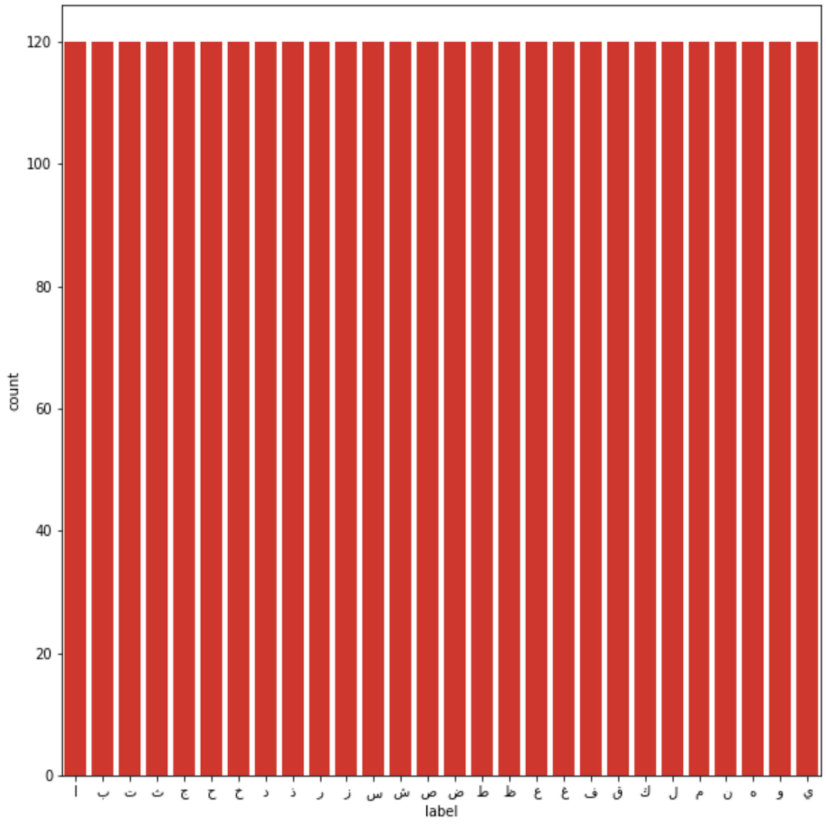

Figure 3 **Testing images per class.**

Full-size DOI: 10.7717/peerjcs.995/fig-3

ranges from 0 to 27 for all 28 classes. Finally, the training and testing sets were reshuffled for increasing the likelihood of the best fitted model.

Convolutional neural network (CNN)

CNN is one of the most commonly used kinds of artificial neural networks (ANN) (*Acharya et al., 2017*) that has shown extraordinary contribution in image related problems (*Altwaijry & Al-Turaiki, 2021*). CNN is typically consisted of three major layers such as convolution, pooling and fully connected layers (*Sultana, Sufian & Dutta, 2018*) and an output layer.

In the convolution layer of a CNN, an image is convolved with filters to create feature maps (*Yu, Jia & Xu, 2017*) which forward these maps to the subsequent layer for extracting another higher-level of feature set from the input image (*Abiwinanda et al., 2019*). In this layer, most of the filters work concurrently on the same input image while each filter identifies a distinct parameter (*Singh & Shankar, 2021*). Furthermore, the common activation function used in this layer addressed non-linearity such as sigmoid, tanh, ReLU, and leaky ReLU (*Yu, Jia & Xu, 2017*).

A pooling layer is normally used between the convolution layers for reducing the dimensionality of an image while keeping the foremost features in the feature maps

(*Abiwinanda et al., 2019*). The dimensionality reduction happens in two ways, either by selecting the maximum region or average region of an image (*Singh & Shankar, 2021*). Moreover, pooling reduces the image size but keeps the predominant features (*Tekerek, 2021*). A flattening layer is used to vectorize the feature maps which is normally occurs right after the last convolution layer. The flatten input vector passed into the fully connected layers which output the predicted class based on the correctly classified samples at each output neuron (*Abiwinanda et al., 2019*). Moreover, the activation function used on the fully connected layer is ReLu, sigmoid, etc., except for the output layer which is softmax that gives the output of the predicted class.

In this study, a hyperparameter tunning was performed using grid parameter tuning. This method suggests all possible combinations of hyperparameters. The optimal combination of the hyperparameters is then selected using the selected best fitted model.

Proposed CNN Model

The proposed CNN model is consisted of a total of eight layers including three convolution layers, three max-pooling layers, and two fully connected layers as shown in Figs. 4 and 5. The input layer takes the input in three dimensions that are height, width, and dimension (h × w × d). The $D = 3$ represents the RGB image while the $D = 1$ shows grey scale image. In the proposed model, the input image was a grey scale image of size 32× 32× 1. In each convolution layer, the layers are convolved with respect to the kernel size of (3×3). The small kernel size of (3×3) is appropriate for this classification process. In each convolution layer, the activation function used was ReLu for non-linearity purposes and padding was used to prevent the image shrinkage.

$$ReLu\ (z) = \max\ (0, z). \tag{1}$$

In CNN, the input layer provides only the image shape without any learning; therefore, the number of parameters is 0. The actual learning of the CNN model starts at the convolution layers. In this model, the input image is provided in a grey scale of shape 32× 32× 1. The first convolution layer used 32 features, kernel size of (3×3), strides ($s = 1$), and same-padding of size 1. The output shape of the layers is calculated as (filters + 2× padding –(kernel_size - 1)). Thus, the output shape of the first layer is 32× 32× 32. The activation size is the dot product of the output shape, resulting in the first layer activation size of 32,768 elements. Table 1 shows the output structures and activation sizes of all layers.

The kernel size (3× 3) is in fact in the form of (h, w, d) *i.e.,* (3× 3× 1). The number of the trainable parameters at each layer is the dot product of h, w, and d. Additionally, a bias is added to each feature to adjust the output alongside the weighted sum of the inputs. The number of trainable parameters at a convolution layer is calculated as ((h * w * d) +1) * k), where k is output features and d is the dimension of the previous layer. A general formula for calculating the number of trainable parameters in each layer using Eq. (2).

$$Z_n = \sum_{i=0}^{j-1} (f_i * x_{n-i}) + b \tag{2}$$

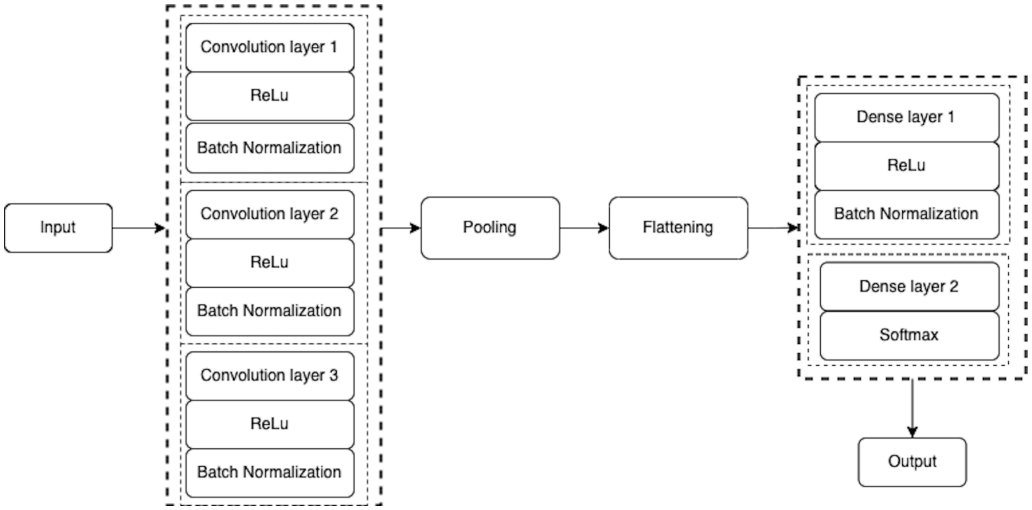

Figure 4 **A schematic architecture of CNN.**

Full-size DOI: 10.7717/peerjcs.995/fig-4

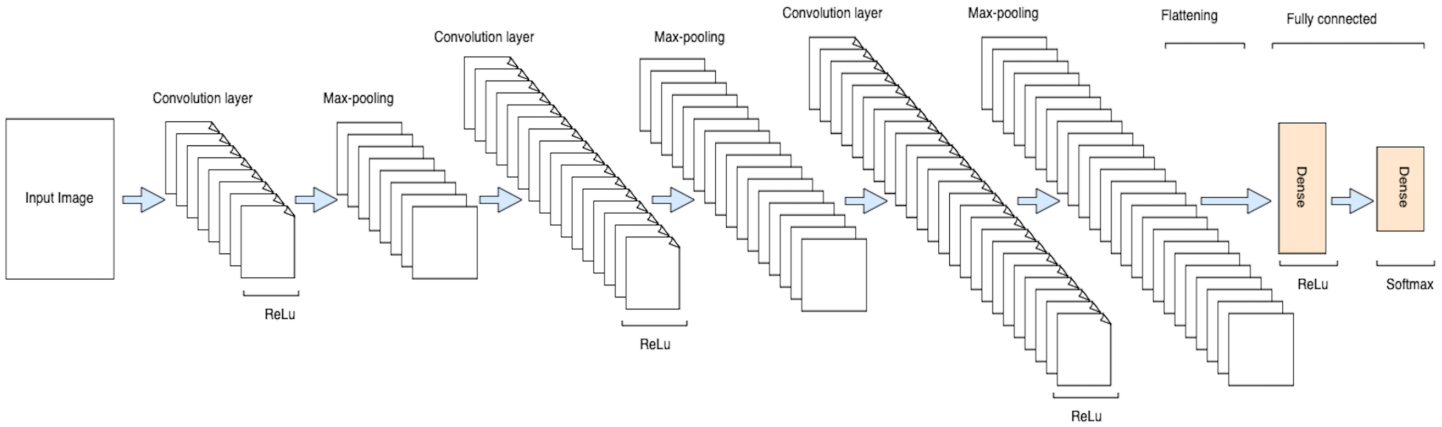

Figure 5 **Architecture of the proposed CNN model.**

Full-size DOI: 10.7717/peerjcs.995/fig-5

where, f is the current kernel size, x is the previous layer output, b is a bias constant, and z is the output vector.

Currently, the trainable parameters in the first convolution layer *i.e.*, $((3 * 3 * 1) + 1) * 32 = 320$. Table 2 shows the number of trainable parameters in each layer of the CNN model.

A max-pooling operation was performed after the convolution layer for reducing the dimensionality of the image. A max-pooling is used to capture the most important features (*Gambäck & Sikdar, 2017*) from the feature maps. In this layer, the maximum number of predominant features are selected from the previous feature map hold by a filter and output the most prevalent features (*GeeksforGeeks, 2022*). In this model, the max-pooling

Table 1 The output structure and size of layers.

No.	Layer	Output shape	Size
1.	Input layer	32, 32, 1	1,024
2.	Convolution 1	32, 32, 32	32,768
3.	Max-pooling1	16, 16, 32	8,192
4	Batch normalization	16, 16, 32	8,192
4.	Convolution 2	16, 16, 64	16,384
5.	Max-pooling 2	8, 8, 64	4,096
6.	Batch normalization	8, 8, 64	4,096
7.	Convolution 3	8, 8, 128	8,192
8.	Max-pooling 3	4, 4, 128	2,048
9.	Batch normalization	4, 4, 128	2,048
10.	Fully connected layer 1	32	—
11.	Fully connected layer 1	28	—

Table 2 CNN layers and trainable parameters.

No.	Layer	Trainable parameters
1.	Input layer	0
2.	Convolution 1	320
3.	Max-pooling1	0
4	Batch normalization	128
4.	Convolution 2	18,496
5.	Max-pooling 2	0
6.	Batch normalization	256
7.	Convolution 3	3,856
8.	Max-pooling 3	0
9.	Batch normalization	512
10.	Fully connected layer 1	65,568
11.	Batch normalization	128
12.	Fully connected layer 1	924
Total trainable parameters		159,676

of slide size (2× 2) was used for dimensionality reduction which formed a layer of size 16× 16× 32 as shown in Table 1.

Moreover, a batch normalization (shown in Eq. (3)) operation was also performed after pooling to normalize the results of the preceding layer which permits independent learning of each layer in the network. According to *Santurkar et al. (2018)* the mechanism of batch normalization aims to make steady the inputs distribution to the network layer in a training time. Similarly, the dropout regularization layer (*Poernomo & Kang, 2018*) was also added for handling the overfitting problem.

$$x^N = \frac{x - mean(x)}{std(x)}. \tag{3}$$

Similarly, 64 features, kernel size (3× 3), same-padding, and $s = 1$, were used in the second convolution layer which resulted an output layer of shape 16× 16× 64 and activation size of 16,384 elements as shown in Table 1. The number of parameters of this layer calculated using Eq. (2) yields a total of 18,496 trainable parameters as shown in Table 2. Again, the activation function used was ReLu as shown in Eq. (1) for handling the linearity problem. After the second convolution, the max-pooling of size (2× 2) was used for dimensionality reduction. Batch normalization and dropout layers were added the same as in the first layer.

In the third convolution, 128 features, kernel size (3× 3), same-padding of size 1, $s = 1$, and ReLu activation function as in Eq. (1) were used. The output shape of the third layer was of shape 8× 8× 128 and activation size of 8192 elements as shown in Table 1. The total number of trainable parameters as per Eq. (2) is 3856 as shown in Table 2. A max-pooling was used for dimensionality reduction and batch normalization was utilized for normalizing the results of the previous layer. At the end, a dropout regularization was used to handle the overfitting problems.

Within the CNN model, after analyzing the convolution layer, the flattening layer is used to vectorize the feature map and pass it to the connected layers. The output of the flatten layer is used as an input for the connected layers. Normally, there is no such special operation performed, but the only thing to do in this layer is to vectorize the feature map. Thus, the flatten vector size achieved was 2048.

The final step in the CNN model is the fully connected layers. This is worth mentioning that fully connected layers are consisted of several layers depends on the problem solution in which each neuron of one layer is connected to the neurons of the subsequent layer (*Altwaijry & Al-Turaiki, 2021*). In this model, the fully connected layer was built of size 32 neurons. The activation function used in the fully connected layer was ReLu as in Eq. (1) except in the output layer where soft-max was used instead as shown in Eq. (4). The fully connected layer was followed by batch normalization and dropout layers. The output layer or soft-max layer consisted of 28 neurons used to classify each letter based on the predicted value.

$$\sigma(Z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}. \tag{4}$$

Moreover, for the optimization of the proposed model, this study has tested three different optimizers: RMSProp, Adam, and Adagrad. The test results in Adam were more promising; therefore, the proposed model was optimized using the Adam optimizer. The categorical cross-entropy is used for measuring loss in predicting multi-class labels.

Subsequently, the model was trained using the Adam optimizer. The training set was divided into several batches, where each batch size consisted of equal number of samples. Moreover, the model was tested using different number epochs. The final optimal epochs were set to 50 and run the experiment on a Google Colab notebook with a GPU environment. The performance of the proposed model was evaluated using various evaluation measures.

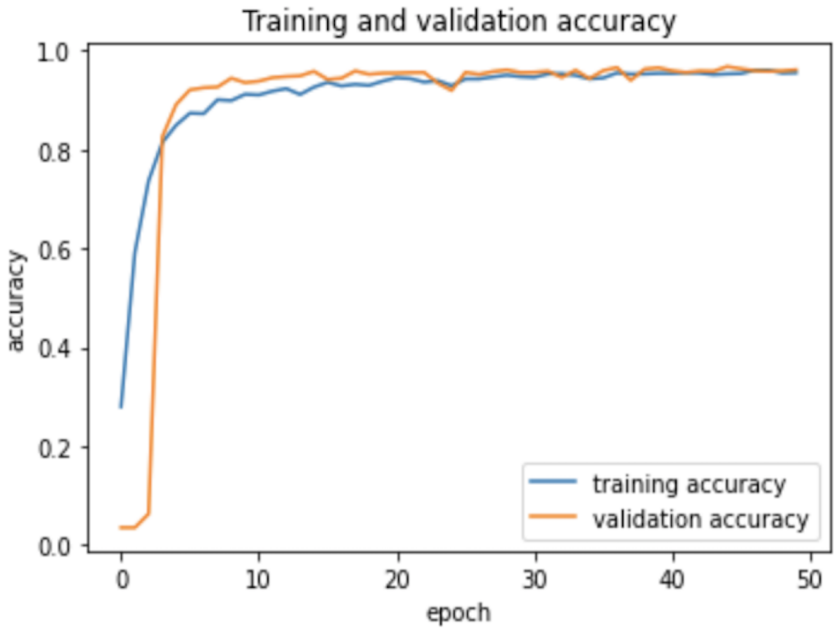

Figure 6 **Learning accuracies with respect to epochs.**
Full-size DOI: 10.7717/peerjcs.995/fig-6

Experiment

As the AHCD dataset is consisted of 16,800 images of which 80% of images contained in the training set, while 20% remained in the testing set. In the experiment of this study, the training set was further divided into two sets: 70% of the images for training and 30% for validation. The model was created using Adam optimizer, the loss was measured using categorical cross-entropy, and the metric was set to accuracy. The model was trained using 2 GHz Quad-Core Intel Core i5 on a Google Colab notebook with a GPU environment. The summarized model shows that the total trainable parameters were 159,676 as shown in Table 2. There were several attempts made with different numbers of epochs to achieve the best fit model; however, the epochs number set to 50 resulted in an optimal model for Arabic handwritten letter recognition. The average time of each epoch during training the model was around 9 s. Figure 6 shows the training and validation accuracies with respect to epochs. Normally, the training accuracy is high but both curves in Fig. 6 are very near to each other. Figure 7 shows the losses of training and validation. From Fig. 7, it has shown the lack of overfitting because as both curves advance, they coincide. Moreover, the overfitting problem was handled using dropout regularization which is supported by several studies (*Baldi & Sadowski, 2013*; *Xu & Liu, 2020*; *Liang et al., 2021*). During training, each node in the training epoch shows a probability parameter that is set by the dropout to be preserved and will be set to 0 otherwise (*Xu & Liu, 2020*).

Therefore, the dropout operation is widely used for preserving the probability parameter of each node in the training epoch, resulting in handling the overfitting problems. Figs. 8 and 9, show the accuracies and losses of training and validation curves respectively when the dropout was not used. In Fig. 8, the validation accuracies have deviated from the training

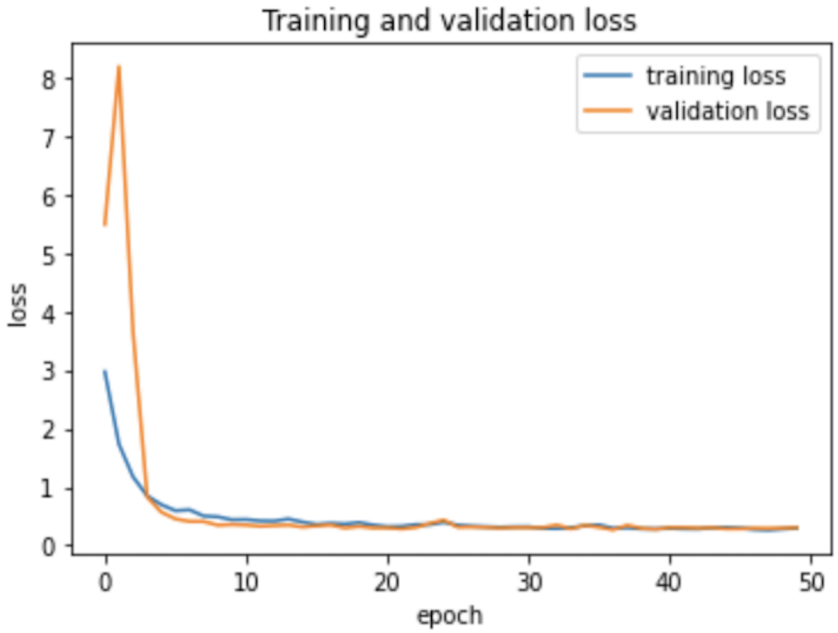

Figure 7 **Learning losses with respect to epochs.**
Full-size DOI: 10.7717/peerjcs.995/fig-7

curve. Similarly, Fig. 9 shows the divergence of the validation curve towards up starting from the first epoch, showing the model is overfitted. Comparing Figs. 9 to 7 above, the proposed model built with dropout regularization is best fitted.

RESULTS AND DISCUSSION

The performance evaluation of the proposed model was measured using accuracy, precision, recall, and f-measure using the following equations.

– The proportion of correctly recognized images to the entire number that is predicted (*Ullah et al., 2021*). Equation (5) shows computes accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

– Precision is the proportion of relevant images among the retrieved images (*Ali et al., 2021*) which is computed using Eq. (6).

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

– A recall is the proportion of relevant images that were retrieved (*Ullah et al., 2021*) which is computed using Eq. (7).

$$Recall\ or\ Sensitivity = \frac{TP}{TP + FN} \tag{7}$$

– The weighted average of precision and recall is known as the f-measure (*Ullah et al., 2021*) which is computed using Eq. (8).

$$\mathrm{F} - measure = \frac{(2 * Precision * Recall)}{Precision + Recall}. \tag{8}$$

Figure 8 Learning accuracies without dropout.

Full-size DOI: 10.7717/peerjcs.995/fig-8

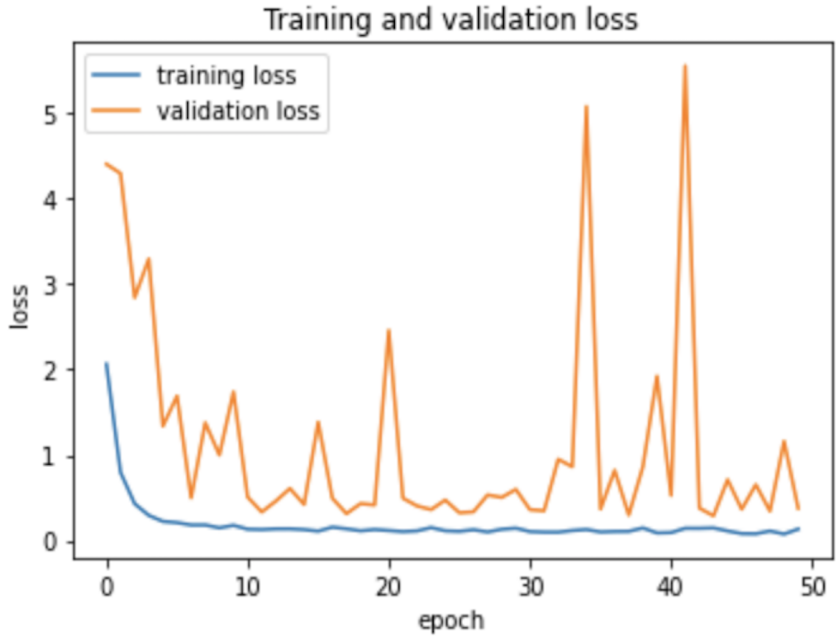

Figure 9 Learning losses without dropout.

Full-size DOI: 10.7717/peerjcs.995/fig-9

Table 3 Model classification report.

Index No.	Class (Arabic letters).	Precision.	Sensitivity.	F-measure
0	أ	0.98	1.00	0.99
1	ب	0.98	0.99	0.99
2	ت	0.97	0.92	0.94
3	ث	0.94	0.96	0.95
4	ج	0.98	0.99	0.99
5	ح	0.97	0.95	0.96
6	خ	0.95	0.97	0.96
7	د	0.96	0.97	0.97
8	ذ	0.92	0.95	0.93
9	ر	0.94	0.99	0.97
10	ز	0.97	0.89	0.93
11	س	0.99	0.98	0.99
12	ش	0.98	1.00	0.99
13	ص	0.96	0.99	0.98
14	ض	1.00	0.93	0.97
15	ط	0.96	0.99	0.98
16	ظ	0.99	0.96	0.97
17	ع	0.98	0.97	0.98
18	غ	0.97	0.97	0.97
19	ف	0.94	0.97	0.95
20	ق	0.93	0.95	0.94
21	ك	0.97	0.97	0.97
22	ل	1.00	1.00	1.00
23	م	1.00	0.97	0.99
24	ن	0.93	0.95	0.94
25	ه	0.97	0.97	0.97
26	و	0.96	0.96	0.96
27	ي	0.99	0.98	0.99
Accuracy				0.97
Macro avg		0.97	0.97	0.97
Weighted avg		0.97	0.97	0.97

The overall accuracy of the proposed model for Arabic letter recognition is 96.78%. The model was also evaluated using other measures, such as precision, recall, and F-measures as shown in Table 3. The precision or confidence of a model is a ratio of correctly classified positive values to the total positive classified values (*Ullah et al., 2021*). The precision or positive predictive value (PPV) is calculated using Eq. (6). Recall or sensitivity of a model is calculated using Eq. (7), which is the ratio of correctly classified positive values to the whole values in the actual class (*Ahmad et al., 2021*). Similarly, F-measure as per Eq. (8) is the weighted average of PPV and sensitivity (*Ullah et al., 2021*; *Al-Mudimigh, Ullah & Alsubaie, 2011*).

Table 4 Comparison of the proposed model.

Study	Model	Dataset	Accuracy
El-Sawy, Loey & EL-Bakry (2017)	CNN	AHCD	94.9%
Proposed Model	CNN	AHCD	**96.78%**

The classification report in Table 3 shows promising results of the proposed model in terms of PPV, sensitivity and f-measures. In the proposed model, the average PPV, sensitivity, and f-measure are 96.71%, 96.75%, and 96.86% respectively. Moreover, individual letter values are shown in Table 3.

Comparing the result of our proposed model with the previous study exhibited in Table 4. *El-Sawy, Loey & EL-Bakry (2017)* have developed a CNN model containing two convolution layers of filter sizes 80 and 64 respectively. The convolution layers are followed by pooling layers. Regularization was used for handling overfitting and a fully connected layer of 1924 neurons was used followed by an output layer of 28 neurons. The model achieved a testing accuracy of 94.9% on the testing set. Our proposed intelligent model developed for recognizing the Arabic handwritten letters outperformed the *El-Sawy, Loey & EL-Bakry (2017)* model. The reported results showed an improvement rate of approximately 2% than the previous study. This enhancement in accuracy was due to using the different number of layers, regularization, number of iterations as well as the preprocessing steps discussed in the early sections. This is to be noted that the AHCD dataset was prepared by *El-Sawy, Loey & EL-Bakry (2017)* and also the first who tested AHCD using CNN. Therefore, we have compared our results with them.

Moreover, the proposed model has been tested on the unseen images and produced amazing results as shown in Fig. 10. In this step, the 10 images were randomly selected from a test set in the AHCD dataset. The proposed model printed the predicted and the actual indices for the 10 randomly selected images as shown in Table 5.

As in the data preprocessing stage, the indices of the images were reshaped that is starting from '0' instead of '1' in order for a better understanding of the classifier; therefore, in this study, the Arabic letters started from '0'. According to this standard indexing, the first letter which is 'ا' (alef) is at index '0' and the last letter which is 'ي' (yaa) at index '27'. As per Table 5, indices 2, 5, 27, 13, 5, 1, 20, 19, 14, and 16 represent 'ت' (taa), 'ح' (haa), 'ي' (yaa), 'ص' (sad), 'ح' (haa), 'ب' (baa), 'ق'(qaf), 'ف' (faa), 'ض' (dad), and 'ظ' (dhaa) respectively. The corresponding test results of the proposed model are shown in Fig. 10. As both the ground reality indices and the predicted indices are the same after testing the unseen images on the proposed model, reports the reliability of the proposed model for recognizing the Arabic handwritten letters.

The Arabic letter recognition is a challenge due to the similar structure of several letters such as a set of {'ب' (baa), 'ت' (taa), 'ث' (thaa)} which can only be differentiated with the dot(s) at the top or bottom of the letter. The positions of these dots are not necessarily to be in the same position with every person because sometimes these dots are written very confusing like three dots in 'ث' are closely similar to the letter 'ت' or the dot of letter 'ب' comes very closer to the letter itself and form a new shape or sometimes the two dot letter

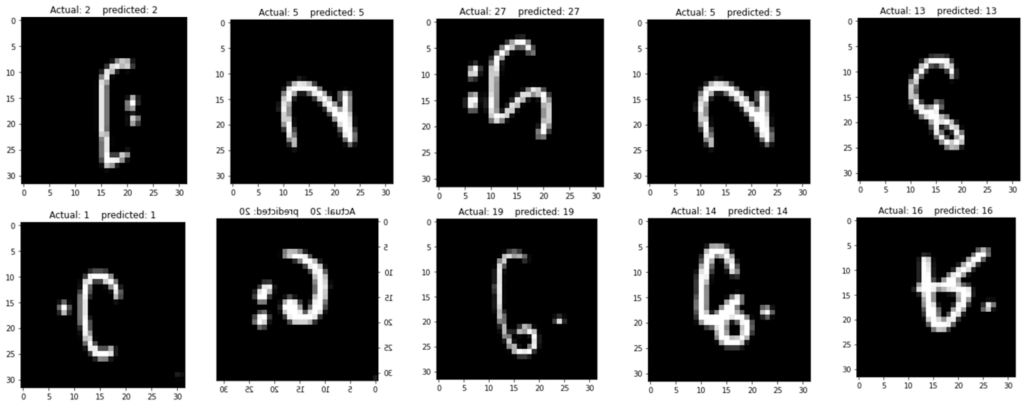

Figure 10 **Model testing on unseen images.**
Full-size DOI: 10.7717/peerjcs.995/fig-10

Table 5 **Model testing on unseen images.**

No. of images	10
Actual indices	2, 5, 27, 5, 13, 1, 20, 19, 14, 16
Predicted indices	2, 5, 27, 5, 13, 1, 20, 19, 14, 16

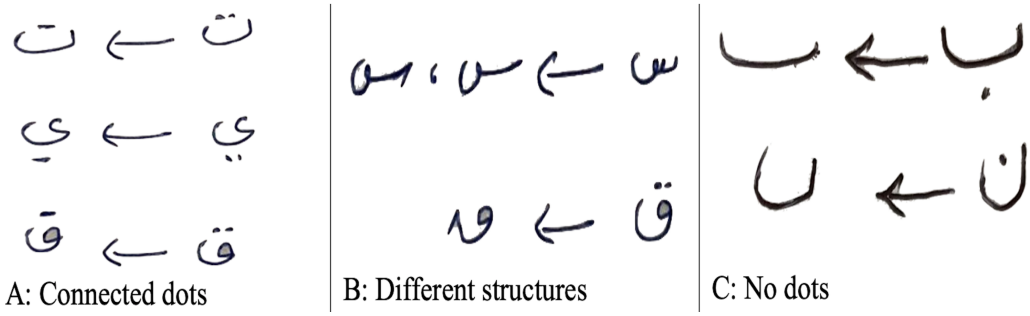

Figure 11 **Letters representation in different forms.**
Full-size DOI: 10.7717/peerjcs.995/fig-11

‘ت’ is written very light, showing just one dot, ultimately resembling the letter ‘ن’(noon). It is worth mentioning that most Arabic native speakers write the two dots connected to each other and form a line (see Fig. 11A) which leads to an ambiguous situation, thus posing a challenge to digital recognition. Similar case for other sets like {‘ج’ (geem), ‘ح’ (haa), ‘خ’ (khaa)}; {‘د’ (dal), ‘ذ’ (thal)}, {‘س’ (sean), ‘ش’ (shean)}, {‘ص’ (sad), ‘ض’ (dad)}, { ‘ط’ (taah), ‘ظ’ (dhaa)}, {‘ع’ (aain), ‘غ’ (ghain)}, etc. These challenges were also reported by *El-Sawy, Loey & EL-Bakry (2017)*.

Another difficulty associated with Arabic letter recognition is the formation of a particular letter depending on an individual’s writing style. For example, the letters ‘س’ (sean) and ‘ق’(qaf) can be written in several forms, as shown in Fig. 11B. There are some letters as shown in Fig. 11C, which are by default understood to human; therefore,

sometimes they write them without dots, but for the classifier, it is very difficult to differentiate and hence, lead to the confusion regarding not only the prediction of the original letters but between differentiating the letters 'ب' and 'ن' which are identical without dots (see Fig. 11C). Besides all these challenges, our proposed model is best fitted on AHCD dataset for recognizing the offline handwritten Arabic letters. Overall, the testing results are promising in terms of letter recognition.

CONCLUSIONS AND FUTURE WORK

This study proposed an intelligent approach based on CNN for recognizing handwritten Arabic letters. In this approach, regularization was utilized and the model overfitting was prevented using dropout operation, and batch normalization was used to normalize the outputs and allow independent learning of each layer in the network. The model was tested with and without dropout and found significant differences in the experimental results as shown in a set of figures (Figs. 6 and 7 *vs.* 8 and 9). In this study, ReLu was used as an activation function in all layers except the softmax layer and the model was created using the Adam optimizer. The experimental results shown in the above tables and figures reported the best fit of the proposed model. Moreover, the model was also tested on unseen samples and achieved excellent results (see Fig. 10). The comparison table shows the enhanced accuracy rate *i.e.,* 96.78% of the proposed model contributes to the reliability of the model best fit for the intelligent recognition of handwritten Arabic letters.

As the handwritten scripts recognition is of active interest to several researchers. Therefore, in the future, the proposed model can be applied to other recognition tasks, such as the recognition of handwritten Arabic connected letters or script, Arabic digits, or handwritten alpha-numerical items occurring in similarly patterned languages, such as Urdu, Persian, Pashto, etc.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R104), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
The Princess Nourah bint Abdulrahman University Researchers Supporting Project number: PNURSP2022R104.
Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Zahid Ullah conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Mona Jamjoom conceived and designed the experiments, performed the experiments, performed the computation work, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The dataset used in this study is publicly available at Kaggle: https://www.kaggle.com/mloey1/ahcd1.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.995#supplemental-information.

REFERENCES

Abdalkafor AS, AlHamouz S. 2016. Arabic offline handwritten isolated character recognition system using neural network. *International Journal of Business and ICT* **2**:41–50.

Abiwinanda N, Hanif M, Hesaputra ST, Handayani A, Mengko TR. 2019. Brain tumor classification using convolutional neural network. In: *World congress on medical physics and biomedical engineering*. Springer Singapore, 183–189.

Acharya UR, Oh SL, Hagiwara Y, Tan JH, Adam M, Gertych A, Tan RS. 2017. A deep convolutional neural network model to classify heartbeats. *Computers in Biology and Medicine* **89**:389–396 DOI 10.1016/j.compbiomed.2017.08.022.

Ahmad H, Ahmad S, Asif M, Rehman M, Alharbi A, Ullah Z. 2021. Evolution-based performance prediction of star cricketers. *Computers, Materials and Continua* **69**:1215–1232 DOI 10.32604/cmc.2021.016659.

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. In: *International conference on brain inspired cognitive systems*. Springer International Publishing, 457–468.

Ahmed R, Gogate M, Tahir A, Dashtipour K, Al-Tamimi B, Hawalah A, El-Affendi MA, Hussain A. 2021. Deep neural network-based contextual recognition of Arabic handwritten scripts. *Entropy* **23**:1–27.

Al-Helali BM, Mahmoud SA. 2017. Arabic online handwriting recognition (AOHR): a survey. *ACM Computing Surveys* **50**:1–35 DOI 10.1145/3060620.

Al-Mudimigh AS, Ullah Z. 2011. Prevention of dirty data and the role of MADAR Project. In: *UKSim 5th european symposium on computer modeling and simulation*. Madrid, Spain.

Al-Mudimigh AS, Ullah Z, Alsubaie TA. 2011. A framework for portal implementation: a case for Saudi organizations. *International Journal of Information Management* **31**:38–43 DOI 10.1016/j.ijinfomgt.2010.05.001.

Al-Taani AT, Al-Haj S. 2010. Recognition of on-line Arabic handwritten characters using structural features. *Journal of Pattern Recognition Research* **5**:23–37 DOI 10.13176/11.217.

Ali AAA, Mallaiah S. 2021. Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout. *Journal of King Saud University - Computer and Information Sciences* DOI 10.1016/j.jksuci.2021.01.012.

Ali AAA, Suresha M. 2019. Arabic handwritten character recognition using machine learning approaches. In: *Fifth international conference on image information processing (ICIIP)*. 187–192.

Ali H, Ullah A, Iqbal T, Khattak S. 2020. Pioneer dataset and automatic recognition of Urdu handwritten characters using a deep autoencoder and convolutional neural network. *SN Applied Sciences* **2**:1–12 DOI 10.1007/s42452-019-1914-1.

Ali MR, Ahmad F, Chaudary MH, Khan ZA, Alqahtani MA, Alqurni JS, Ullah Z, Khan WU. 2021. Petri Net based modeling and analysis for improved resource utilization in cloud computing. *PeerJ Computer Science* **7**:1–22.

Aljarrah MN, Zyout MM, Duwairi R. 2021. Arabic handwritten characters recognition using convolutional neural network. In: *12th international conference on information and communication systems, ICICS 2021*. IEEE, 182–188.

Almansari OA, Hashim NNWN. 2019. Recognition of isolated handwritten Arabic characters. In: *7th international conference on mechatronics engineering, ICOM 2019*. IEEE, 1–5.

Altwaijry N, Al-Turaiki I. 2021. Arabic handwriting recognition system using convolutional neural network. *Neural Computing and Applications* **33**:2249–2261 DOI 10.1007/s00521-020-05070-8.

Bai J, Chen Z, Feng B, Xu B. 2014. Image character recognition using deep convolutional neural network learned from different languages. In: *2014 IEEE international conference on image processing, ICIP 2014*. 2560–2564.

Balaha HM, Ali HA, Badawy M. 2021. Automatic recognition of handwritten Arabic characters: a comprehensive review. *Neural Computing and Applications* **33**:3011–3034 DOI 10.1007/s00521-020-05137-6.

Balaha HM, Ali HA, Youssef EK, Elsayed AE, Samak RA, Abdelhaleem MS, Tolba MM, Shehata MR, Mahmoud MR, Abdelhameed MM, Mohammed MM. 2021. Recognizing Arabic handwritten characters using deep learning and genetic algorithms. *Multimedia Tools and Applications* **80**:32473–32509 DOI 10.1007/s11042-021-11185-4.

Baldi P, Sadowski P. 2013. Understanding dropout. *Advances in Neural Information Processing Systems*.

Boufenar C, Kerboua A, Batouche M. 2018. Investigation on deep learning for off-line handwritten Arabic character recognition. *Cognitive Systems Research* **50**:180–195 DOI 10.1016/j.cogsys.2017.11.002.

Djaghbellou S, Akhtar Z, Bouziane A, Attia A. 2020. Arabic handwritten characters recognition via multi-scale hog features and multi-layer deep rule-based classification. *Journal on Image and Video Processing* **10**:2195–2200.

El-Sawy A, Loey M, EL-Bakry H. 2017. Arabic handwritten characters recognition using convolutional neural network. *Wseas Transactions on Computer Research* **5**:11–19.

Elleuch M, Kherallah M. 2018. An improved Arabic handwritten recognition system using deep support vector machines. In: *Computer vision: concepts, methodologies, tools, and applications*. IGI global, 656–678.

Elleuch M, Maalej R, Kherallah M. 2016. A new design based-SVM of the CNN classifier architecture with dropout for offline Arabic handwritten recognition. *Procedia Computer Science* **80**:1712–1723 DOI 10.1016/j.procs.2016.05.512.

Gambäck B, Sikdar UK. 2017. Using convolutional neural networks to classify hate-speech. In: *Proceedings ofthe first workshop on abusive language online*. Vancouver, Canada, 85–90.

GeeksforGeeks. 2022. CNN | Introduction to pooling layer. *Available at https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/* (accessed on 05 January 2022).

Ghosh S, Das N, Nasipuri M. 2019. Reshaping inputs for convolutional neural network: some common and uncommon methods. *Pattern Recognition* **93**:79–94 DOI 10.1016/j.patcog.2019.04.009.

Javed F. 2013. Arabic and English phonetics: a comparative Study. *The Criterion an International Journal in English* **4**:1–13.

Liang X, Wu L, Li J, Wang Y, Meng Q, Qin T, Chen W, Zhang M, Liu T-Y. 2021. R-Drop: regularized dropout for neural networks. In: *35th conference on neural information processing systems (NeurIPS 2021)*. Sydney, Australia, 1–16.

Mudhsh MA, Almodfer R. 2017. Arabic handwritten alphanumeric character recognition using very deep neural network. *Information* **8**:1–14.

Mustapha IB, Hasan S, Nabus H, Shamsuddin SM. 2022. Conditional deep convolutional generative adversarial networks for isolated handwritten Arabic character generation. *Arabian Journal for Science and Engineering* **47**:1309–1320 DOI 10.1007/s13369-021-05796-0.

Poernomo A, Kang DK. 2018. Biased dropout and crossmap dropout: learning towards effective dropout regularization in convolutional neural network. *Neural Networks* **104**:60–67 DOI 10.1016/j.neunet.2018.03.016.

Sahloul A, Suen C. 2014. Off-line system for the recognition of handwritten Arabic character. In: *Fourth international conference on computer science & information technology*. 227–244.

Santurkar S, Tsipras D, Ilyas A, Madry A. 2018. How does batch normalization help optimization? In: *Conference on neural information processing systems*. Montréal, Canada, 1–11.

Shams M, Elsonbaty AA, El Sawy WZ. 2020. Arabic handwritten character recognition based on convolution neural networks and support vector machine. *International Journal of Advanced Computer Science and Applications* **11**:144–149.

Singh P, Shankar A. 2021. A novel optical image denoising technique using convolutional neural network and anisotropic diffusion for real-time surveillance applications. *Journal of Real-Time Image Processing* **18**:1711–1728 DOI 10.1007/s11554-020-01060-0.

Sultana F, Sufian A, Dutta P. 2018. Advancements in image classification using convolutional neural network. In: *4th IEEE international conference on research in computational intelligence and communication networks, ICRCICN 2018*. IEEE, 122–129.

Tekerek A. 2021. A novel architecture for web-based attack detection using convolutional neural network. *Computers and Security* **100**:1–12.

Ullah Z, Al-Mudimigh AS. 2012. Integration and communication to prevent dirty data: the role of MADAR project. *Information* **15**:3459–3467.

Ullah Z, Jamjoom M. 2022. Early detection and diagnosis of chronic kidney disease based on selected predominant features. *Journal of Healthcare Engineering*. In Press.

Ullah Z, Saleem F, Jamjoom M, Fakieh B. 2021. Reliable prediction models based on enriched data for identifying the mode of childbirth by using machine learning methods: Development study. *Journal of Medical Internet Research* **23**:1–12 DOI 10.2196/28856.

Xiao S, Peng L, Yan R, Wang S. 2020. Deep network with pixel-level rectification and robust training for handwriting recognition. *SN Computer Science* **1**:1–13 DOI 10.1007/s42979-020-00133-y.

Xiao X, Jin L, Yang Y, Yang W, Sun J, Chang T. 2017. Building fast and compact convolutional neural networks for offline handwritten Chinese character recognition. *Pattern Recognition* **72**:72–81 DOI 10.1016/j.patcog.2017.06.032.

Xu X, Liu H. 2020. ECG heartbeat classification using convolutional neural networks. *IEEE Access* **8**:8614–8619 DOI 10.1109/ACCESS.2020.2964749.

Younis K. 2017. Arabic hand-written character recognition based on deep convolutional neural networks. *Jordanian Journal of Computers and Information Technology* **3**:186–200 DOI 10.5455/jjcit.71-1498142206.

Yu S, Jia S, Xu C. 2017. Convolutional neural networks for hyperspectral image classification. *Neurocomputing* **219**:88–98 DOI 10.1016/j.neucom.2016.09.010.

Yuan A, Bai G, Jiao L, Liu Y. 2012. Offline handwritten English character recognition based on convolutional neural network. In: *10th IAPR international workshop on document analysis systems, DAS 2012*. 125–129.

Zhong Z, Jin L, Feng Z. 2015. Multi-font printed Chinese character recognition using multi-pooling convolutional neural network. In: *13th international conference on document analysis and recognition, ICDAR*. IEEE, 96–100.