Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05397-2

ORIGINAL ARTICLE

A new Arabic handwritten character recognition deep learning system (AHCR-DLS)

Hossam Magdy Balaha[1] · Hesham Arafat Ali[1] · Mohamed Saraya[1] · Mahmoud Badawy[1,2]

Received: 4 March 2020 / Accepted: 24 September 2020

Abstract

Optical character recognition for the English text may be considered one of the most important research topics, whether, printed or handwritten. Although excellent results have been reached in the English text, there is a lack of this type of research in the Arabic text. This is because of the nature of the Arabic alphabet, and the multiplicity of forms of the same letter. Arabic handwritten character recognition (AHCR) systems involve several issues, and challenges from finding a suitable, and public Arabic handwritten text dataset phase to recognition, and classification phase passing through segmentation, and feature extraction phases. The paper objectives are: Firstly, a large, and complex Arabic handwritten characters' dataset (HMBD) is presented for training, testing, and validation phases, as well as, discussing its collection, preparation, cleaning, and preprocessing. Secondly, we introduce a deep learning (DL) system with two convolutional neural network (CNN) architectures (named HMB1 and HMB2); with the appliance of optimization, regularization, and dropout techniques. This system can serve as a baseline for future research on handwritten Arabic text. Different performance metrics were calculated such as accuracy, recall, precision, and F1. 16 experiments were applied to the described system using HMBD, and another two datasets: CMATER, and AIA9k. Experiments' results were captured and compared to study the effects of weight initializers, optimizers, data augmentation, and regularization on overfitting, and accuracy. He Uniform weight initializer and AdaDelta optimizer reported the highest accuracies. Data augmentation showed an improvement in the accuracies. HMB1 reported testing accuracy of 98.4% with 865,840 records using augmentation on HMBD. CMATER and AIA9k datasets were used for validating the generalization. Data augmentation was applied, and the best results were 100%, and 99.0% for testing accuracies, respectively. A cross-over validation between the described architectures, and a previous state-of-the-art architecture, and dataset was performed in two phases. First, the previous control architecture cannot generalize for the presented dataset in the current study. Second, the study described architectures generalize for the control dataset, with higher accuracies (97.3%, and 96.8% for HMB1, and HMB2, respectively), than the reported accuracy in the selected control study.

Keywords Arabic handwritten character recognition · Classification · Convolutional neural network · Data augmentation · Deep learning · Optical character recognition · Optimizers

✉ Hossam Magdy Balaha
hossam.m.balaha@mans.edu.eg

Hesham Arafat Ali
h_arafat_ali@mans.edu.eg

Mohamed Saraya
mohamedsabry83@mans.edu.eg

Mahmoud Badawy
engbadawy@mans.edu.eg

[1] Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt

[2] Department of Computer Science and Informatics, Taibah University, Al Medina Al Munawara, Saudi Arabia

Abbreviations

AdaDelta	An adaptive learning rate method
Adam	A method for stochastic optimization
AHCR	Arabic handwritten character recognition
AHCR-DLS	Arabic handwritten character recognition deep learning system
CNN	Convolutional neural network
DL	Deep learning
ReLU	Rectified linear unit
SGD	Stochastic gradient descent
UN	United Nations

Published online: 23 October 2020

1 Introduction

Arabic "العربية" language is the official language of 26 countries, and as many as 420 million people around the world speak Arabic, making it the sixth most commonly spoken language [1]. It is also one of the six official languages of the United Nations (UN) joining Chinese, English, French, Russian, and Spanish. In addition, some languages such as Farsi, Kurdish, Urdu, and Pashto, use some of the Arabic words, features, and structures. The Arabic language has 28 characters; which, are written from right to left. The character size is not fixed; rather, it varies according to the shape, font, and position in the word; beginning, middle, end, or isolated. A single character may contain dots (from one to three) or special symbols such as "Hamza", and "Tanween." A single word in English may have different synonyms in Arabic such as the words; "Good", and "Love." That said, represents a major challenge for the researchers in the field of Arabic language understanding [2–6].

Pattern recognition is a major research field that comprises the process of identifying and recognizing different means of input; such as images or streams for various fields. The character, face, and speech recognition are common variants of pattern recognition [7, 8]. One of the pattern recognition steps involves the study of the object; in order to, identify attributes (features), and extract the difference; in other words, to determine matching or mismatching [9]. A significant advancement in the recognition process was achieved with the integration of deep learning (DL) approaches. DL is one of the classes of machine learning, which is based on the artificial neural network. DL is used in detection [10], classification [11, 12], and learning [10, 13, 14]. DL architectures are based on the automatic learning from the features without prior determination (extraction). DL uses many layers to get the final information from the raw data. Many applications in the health risk assessment, drug discovery, computer sciences, face, and sound recognition, medical image analysis, bioinformatics, and robotics adopt the image, and pattern recognition technologies, with the different DL architectures [15, 16]. Some of the frequently used DL classes are deep belief networks, convolutional neural networks (CNNs), recurrent neural networks, and stacked auto-encoders.

CNN is widely used in analyzing, segmenting, and classifying visual imagery. It uses the mathematical operation, convolutions, executed in parallel. CNN consists of one or more convolutional layers with one or more fully connected layers. Compared to the fully connected artificial neural network, CNN is easier, and faster to train and has fewer parameters. After training, CNNs converge and generate acceptable results, and accurate decisions [17]. Arabic handwritten character recognition (AHCR) is considered one of the difficult tasks because of the different writers' styles, ages, and knowledge which affect the shape of the characters, and words [18]. AHCR faces great challenges from different character positions to ligatures [19] passing through special, and punctuations characters [20]. Notwithstanding, a number of studies on AHCR were successful [21–23].

This paper introduces a new method for AHCR using deep learning. It has become recommended [24–26] to train deep neural networks because of the availability of huge amounts of data, and various algorithmic innovations that are taking place. However, for some datasets, the deep learning classification methods need adjustments in their structure and parameters. Arabic characters have certain requirements for processing, and network weights of the neural network to overcome the challenges such as the cursive nature of Arabic language. In this paper, we first introduce a system based on the characteristics of a DL, CNN. Then, we describe how to process the data and construct the dataset.

The paper is organized as follows: in the next section, we describe the previous related AHCR work. In Sect. 3, we discuss the research problem and aims of the work. In Sect. 4, we discuss the methodology from the described system, and data collection to the experimental environment and different validations passing through the data production, processing, and representation. In Sect. 5, we report the experimental results and discussion of these results. Finally, in Sect. 6, we provide a conclusion on this study.

2 Related work

AHCR has many challenges related to the pattern recognition approach in computer science. There are few large Arabic handwritten datasets compared to those in English. Table 1 lists the available Arabic databases that can be used in character recognition.

There are many challenges for researchers to work on in this field, and there is a demand for new methods to emerge as the computational technology is increasing and resource limitations are decreasing [30]. Figure 1 summarizes some of these trending challenges. **Arabic Handwritten Datasets**: Some of the available datasets have not many records as shown in Table 1. There is a need for a dataset with a large amount of data with different font sizes, styles, illuminations, users, and words. **Noise Manipulation**: Images that contain unrequired data noise should be eliminated automatically or manually or it may lead to misclassification. Some of the works depended on removing them

Table 1 Arabic handwritten character databases

Database	Website (if available)	Data type	Dataset size
IFN/ENIT	http://www.ifnenit.com/	Words	26,459
ADBase MADBase	http://datacenter.aucegypt.edu/shazeem/	Numbers	70,000
KHATT	http://khatt.ideas2serve.net/index.php	Text	4000
APTI	https://diuf.unifr.ch/main/diva/APTI/index.html/	Words	45,313,600
AIA9K [27]	http://www.eng.alexu.edu.e.g./~mehussein/AIA9k/index.html	Letters	8737
AlexU-Word	http://www.eng.alexu.edu.e.g./mehussein/alexu-word/index.html	Words	25,114
AHCD [24, 28]	https://www.kaggle.com/mloey1/ahcd1	Letters	16,800
CENPARMI [29]	–	–	21,426
CMATERDB v.3.3.1	https://code.google.com/archive/p/cmaterdb/downloads	Numbers	3000

manually. There is a need to find an easy and fast way to remove them automatically especially for large datasets. **State-of-the-Art Architectures**: Some of the works depended on unsupervised learning and some of them depended on non-state-of-the-art techniques. It is worth to mention that although they achieved low accuracy values compared to the state-of-the-art approaches and architectures. The state-of-the-art supervised techniques provide more scalable and high accuracy to different datasets compared to other techniques. There is a need to depend on these techniques in future researches. **Low-Quality Documents**: Ancient papers and documents are a great challenge as they contain unrequired factors such as noise and some of the characters are removed. It is an open area problem for researchers to work on. **Segmentation**: Some studies depended on the manual segmentation and feature

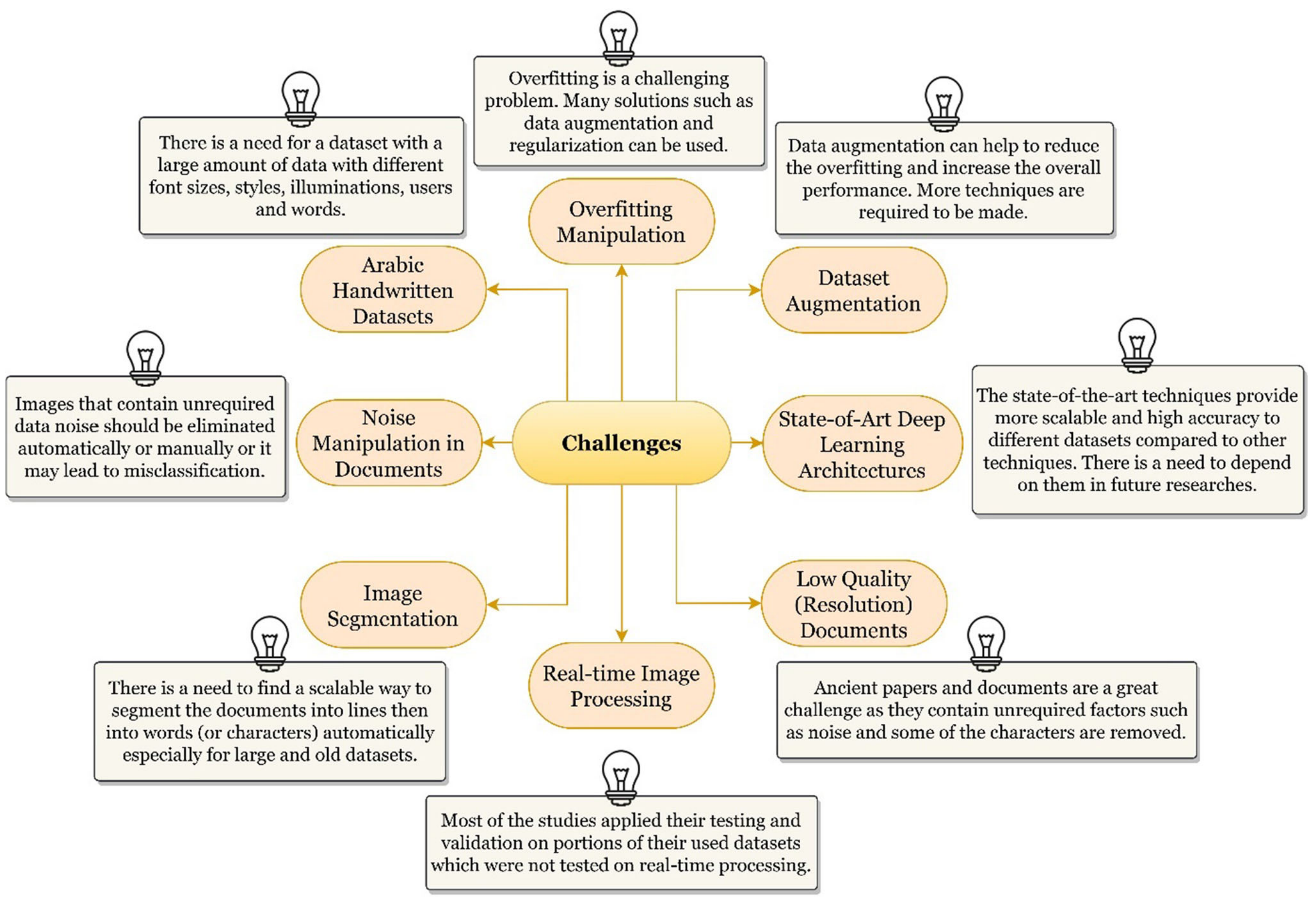


Fig. 1 Trending AHCR Challenges

Table 2 Recent research summarization

Study	Year	Recognition accuracy	Dataset type	Dataset name
El-Sawy et al. [24]	2017	94.9%	Arabic characters	Their own dataset
Younis et al. [25, 39]	2017	94.8%	Arabic characters	AIA9k
		97.6%		AHCD
El-Melegy et al. [26]	2019	97.80%	Arabic literal amounts	Their Own Dataset
Hamida et al. [36]	2019	99.22% for characters	English characters and Arabic numerals	MNIST [40]
		99.74% for digits		
Ashiquzzaman et al. [9]	2019	99.40%	Eastern Arabic numerals	CMATER
Neri et al. [37]	2020	Above 98%	Arabic numerals	Their Own Dataset

extraction of the datasets. There is a need to find a scalable way to segment the documents (pages) into lines then into words (or characters) automatically especially for large and old datasets. **Real-time Image Processing**: Most of the studies applied their testing and validation on portions of their used datasets which were not tested on real-time processing [31–33].

From Fig. 1, the current study works on: (i) Arabic handwritten datasets, (ii) Overfitting manipulation, (iii) Dataset augmentation, (iv) State-of-the-art deep learning architectures, and (v) Noise manipulation in documents. The rest of them can be handled in further studies.

2.1 Recent research in AHCR

Recently, there have been many researches in the field of handwritten recognition in general and Arabic handwritten recognition in particular. A number of studies, based on different datasets, used different tools and methods to facilitate the recognition of the handwritten texts.

El-Sawy et al. [24] described a DL architecture for AHCR that consisted of an input layer, two convolutional layers, two max-pooling layers, a fully connected layer, and an output layer. Their CNN architecture was trained and tested on their dataset that contained 16,800 handwritten Arabic characters. They used only the 28 basic Arabic characters and did not include the different shapes of the Arabic characters nor the digits. They applied different optimization techniques such as regularization and dropout. Their described CNN architecture had an average 5.1% misclassification error on their testing data. El-Melegy et al. [26] investigated a DL architecture for Arabic handwritten literal amounts recognition that consisted of seventeen layers as follows: input layers, three convolutional layers, three batch normalization layers, three rectified linear unit (ReLU) activation layers, three max-pooling layers, three fully connected layers, and an output layer. They focused on the recognition of handwritten Arabic literal amounts with a limited lexicon. They compared their architecture with traditional methods. They applied data augmentation [34, 35] and reached a 97.80% recognition rate.

Two architectures for offline handwritten character recognition were described by Hamida et al. [36]. One of them was a DL architecture that consisted of: an input layer, two convolutional layers (20 filters of 5×5 size), two pooling layers (4×4 size), two fully connected layers, and an output layer. They reached an accuracy of 99.22% for characters and 99.74% for digits. They used the ReLU activation function for the hidden layers and SoftMax for the output layer. Neri et al. [37] described a CNN architecture for handwritten digit recognition. Their experiments showed that their described preprocessing technique led to an accuracy above 98%, which was better than the accuracy obtained with the dataset without the additional preprocessing. They created their datasets with a size of 28×28. Their described architecture consisted of: an input layer, three convolutional layers, two max-pooling layers, one fully connected layer and an output layer. They used ReLU and SoftMax activation functions for the hidden and output layers, respectively.

Another study described a recognition architecture for handwritten Arabic numerals using CNN with the help of data augmentation and dropout techniques [9]. Their architecture reached an accuracy of 99.40%. They trained their architecture on the CMATERDB v.3.3.1 Arabic handwritten digit dataset. The described architecture consisted of an input layer, four convolutional layers, one max-pooling layer, size dropout layers, three connected layers, and an output layer. They used the exponential linear unit (ELU) activation function [38] in the hidden layers. Younis et al. [25, 39] described a DL network for the AHCR problem that uses CNN architectures with regularization parameters such as batch normalization to prevent overfitting. They applied their CNN on the AIA9k [27] and the AHCD [24, 28] datasets, and the classification accuracies for the two datasets were 94.8% and 97.6%, respectively. Table 2 summarizes the recent researches in the field of handwritten recognition. They are sorted from the oldest to the newest.

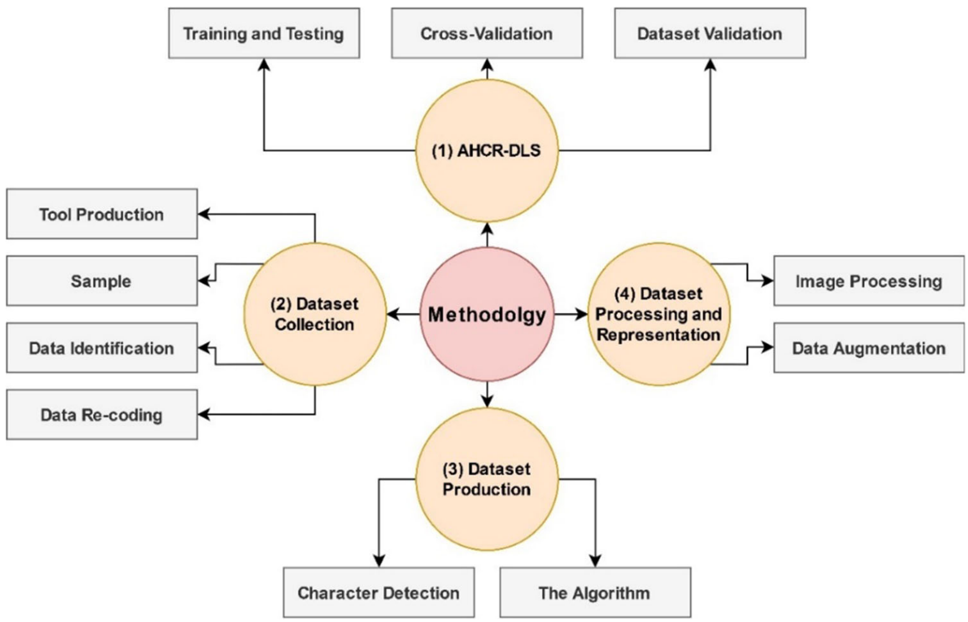


Fig. 2 Current study four phases

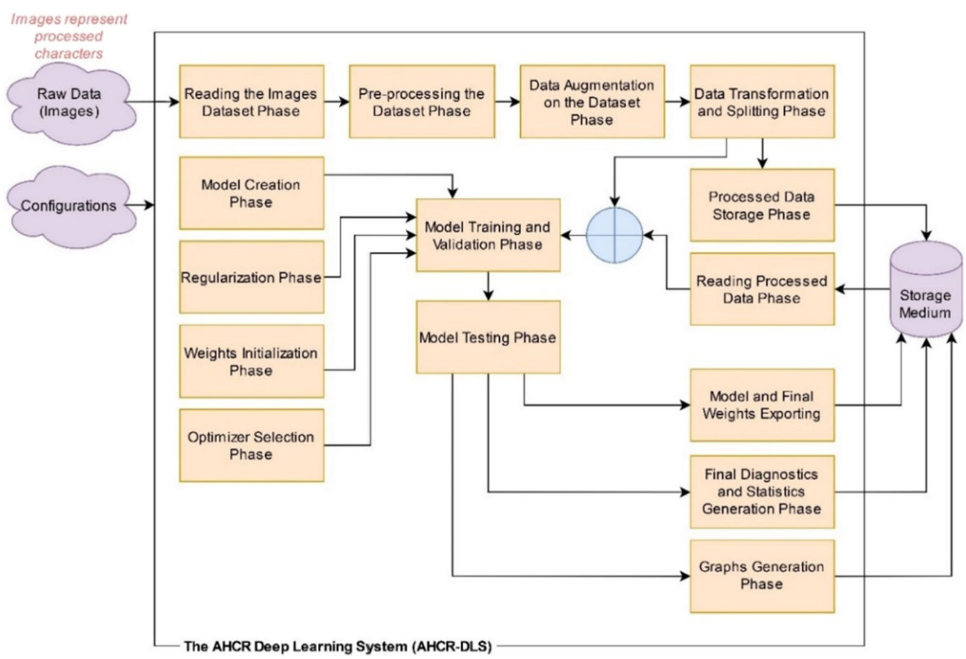


Fig. 3 The AHCR deep learning system (AHCR-DLS)

3 Problem formulation and plan of solution

As seen in Sect. 2, numerous AHCR systems have been described; but, most of them faced great challenges from finding a suitable and public dataset for segmentation and feature extraction, to the recognition and classification. The problem of different character positions, ligatures, diacritics, and punctuations characters mandates the presence of an inclusive and comprehensive dataset to cover all the potential combinations.

Previous studies showed promising results, but the datasets were small and not comprehensive. The quality was high; however, it was not tested in such circumstances, of large combinations. They also did not present a complete AHCR system from collecting the data to classifying it. Actually, these two issues have not been adequately addressed in the literature.

The objectives of this study are to (i) present an AHCR deep learning system (AHCR-DLS) with the ability to select between two CNN architectures, (ii) prepare and design a large and complex dataset (HMBD) for AHCR problems (challenges) as previous datasets are not complex enough, (iii) evaluate the architectures' results by applying different experiments using the described two architectures

Algorithm 1: The AHCR Deep Learning System (AHCR-DLS) Steps

Data: Images Dataset, Configurations
Result: Model, Results, Graphs
$images = readAllImages(ImagesDataset);$
$images - preprocessImages(images, Configurations);$
$images = augmentImages(images, Configurations);$
$train, test, validation = splitImages(images, Configurations);$
$storeSplitImages(train, test, validation);$
$regularizer = selectRegularizer(Configurations);$
$initializer = selectWeightInitializer(Configurations);$
$optimizer = selectOptimizer(Configurations);$
$model = selectConfigModel(Configurations, regularizer, initializer, optimizer);$
$trainedModel = trainModel(model, train, validation);$
$Results = testModel(trainedModel, test);$
$Model = exportModel(trainedModel);$
$Graphs = generateGraphs(trainedModel, Results);$

Fig. 4 The AHCR deep learning system (AHCR-DLS) steps

Table 3 The first CNN architecture (HMB1)

layer type	Output shape	Layer type	Output shape
2D Convolutional Layer	(32, 32, 16)	Batch Normalization	(8, 8, 256)
2D Convolutional Layer	(32, 32, 16)	2D Max-Pooling Layer	(4, 4, 256)
Batch Normalization	(32, 32, 16)	Flatten Layer	(4096)
2D Convolutional Layer	(32, 32, 32)	Dense Layer	(1024)
2D Convolutional Layer	(32, 32, 32)	Batch Normalization	(1024)
Batch Normalization	(32, 32, 32)	Dropout Layer	(1024)
2D Max-Pooling Layer	(16, 16, 32)	Dense Layer	(512)
2D Convolutional Layer	(16, 16, 64)	Batch Normalization	(512)
2D Convolutional Layer	(16, 16, 64)	Dropout Layer	(512)
Batch Normalization	(16, 16, 64)	Dense Layer	(256)
2D Convolutional Layer	(16, 16, 128)	Batch Normalization	(256)
2D Convolutional Layer	(16, 16, 128)	Dropout Layer	(256)
Batch Normalization	(16, 16, 128)	Dense Layer	(128)
2D Max-Pooling Layer	(8, 8, 128)	Batch Normalization	(128)
2D Convolutional Layer	(8, 8, 256)	Dropout Layer	(128)
2D Convolutional Layer	(8, 8, 256)	Dense Layer	(115)

and different datasets including the presented dataset, HMBD, (iv) study the effects of weight initializers, optimizers, data augmentation, and regularization on overfitting and accuracy and finally (v) cross-over validate the overall described work with one of the previous well-known architectures and datasets.

4 Methodology

As shown in Fig. 2, the study includes four phases; First: the AHCR-DLS, second: Data Collection, third: Dataset Production and fourth: Dataset Processing and Representation. The first phase can be used for training, testing, and cross-validation appliance on both the dataset and the architecture. In the next subsections, there will be a detailed description of the four phases of the described methodology.

4.1 AHCR-DLS structure

The system provides an overview of the most trending techniques such as data augmentation and regularization, that can be applied in AHCR with the ability to apply or bypass (if not required). It can be used with different datasets: processed and non-processed datasets and can be used also in transfer learning [41, 42]. As shown in Fig. 3 and Algorithm 1 (Fig. 4), the inputs to the system are the dataset and configurations. The input dataset for the presented system is well-prepared (and will be discussed in a following section), so that, there is no need to further processing; hence, the preprocessing phase can be

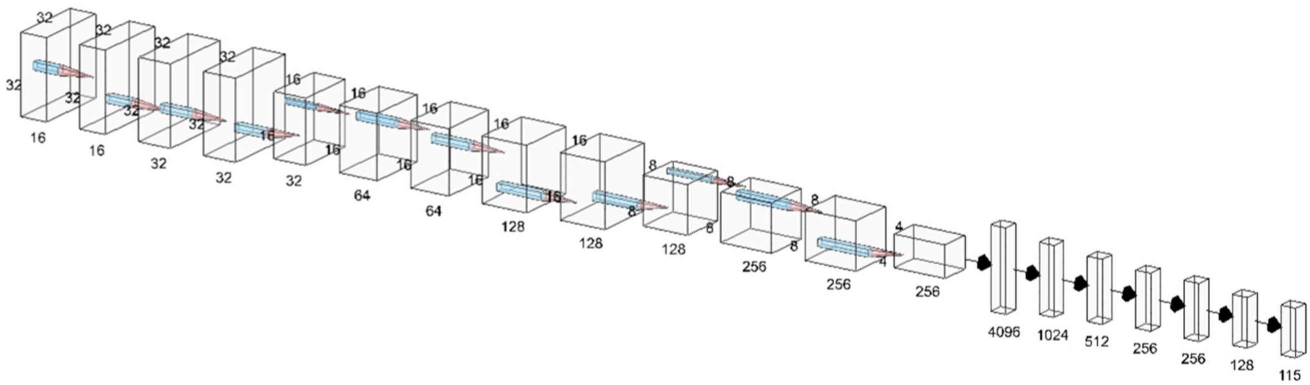


Fig. 5 The first CNN architecture (HMB1)—visual presentation

bypassed. However, it is included in the system design, if a further preprocessing is required. Data augmentation can also be applied as well if required. The dataset is split into training, testing, and validation subsets. Training and validation sets will be used to update the weights, and the test set will be used to measure the architecture performance.

4.1.1 The presented two architectures

The creation phase generates the CNN architectures [43] (discussed separately in the following two paragraphs). They are not used at the same time, but only one is selected. The features are automatically extracted by the used architecture; hence, there is no need to extract the features manually as done in traditional machine learning algorithms. The input layer of all of the architectures is fed to a set of hidden layers and finally to an output layer. The CNN is utilized in the two described architectures of the dataset. The differences between the two described architectures are in the design of the hidden layers; such as the number and type of layers, the number and the size of kernels, and the size of strides. The stride is the number of move pixels that will be applied to the kernel. This will lead to differences in the number of trainable parameters. Weight finalizer, regularization method, and optimizer are selected according to the predefined configurations and will be discussed in a following section. Final output weights, statistics, and graphs are exported and stored.

4.1.2 First CNN architecture (HMB1)

Table 3 and Fig. 5 show a summary of the design of the first described CNN architecture. The 2D convolutional layers have a kernel size of $(3, 3)$ and padding of a type "same." The 2D max-pooling layers have a pooling size of $(2, 2)$. The dropout layers have a dropout ratio of 0.25 and are applied to avoid overfitting and reach better generalization [44]. Batch normalization is applied to stabilize the learning process and to reduce the internal covariate shift [45].

CNNs apply different filters in the whole original image called a convolutional layer. Each filter will represent a single specific image feature. Pooling layers can be used after the convolution layers to reduce the dimensionality and keep the most important features from the previous layer according to the type of the pooling layers such as:

Table 4 The second CNN architecture (HMB2)

Layer type	Output shape	Layer type	Output shape
2D convolutional layer	$(32, 32, 32)$	2D max-pooling layer	$(8, 8, 64)$
Batch normalization	$(32, 32, 32)$	Dropout layer	$(8, 8, 64)$
2D convolutional layer	$(32, 32, 32)$	2D convolutional layer	$(8, 8, 128)$
Batch normalization	$(32, 32, 32)$	Batch normalization	$(8, 8, 128)$
2D Max-Pooling Layer	$(16, 16, 32)$	2D convolutional layer	$(8, 8, 128)$
Dropout layer	$(16, 16, 32)$	Batch normalization	$(8, 8, 128)$
2D convolutional layer	$(16, 16, 64)$	2D max-pooling layer	$(4, 4, 128)$
Batch normalization	$(16, 16, 64)$	Dropout layer	$(4, 4, 128)$
2D convolutional layer	$(16, 16, 64)$	Flatten layer	(2048)
Batch normalization	$(16, 16, 64)$	Dense layer	(115)

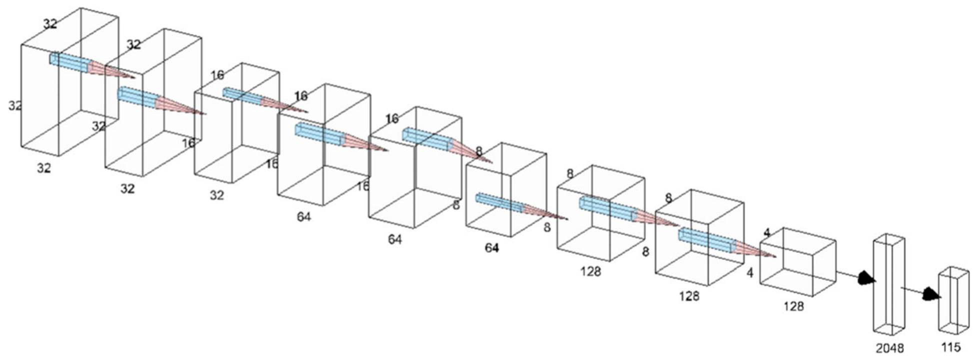


Fig. 6 The Second CNN architecture (HMB2): visual presentation

maximum, average and sum pooling. The last layer is a fully connected layer that uses a one-dimensional vector instead of the two-dimensional arrays used in the preceding layers. The last layer output is the selected class with the highest probability [46]. The 2D convolutional and max-pooling layer output shapes are calculated by Eqs. 1 and 2 [47, 48]:

$$O_{\text{conv}} = \frac{I - k + 2 * p}{s} + 1 \tag{1}$$

$$O_{\text{pooling}} = \frac{I - p_s}{s} + 1 \tag{2}$$

where I is the input shape, k is the kernel size, p is the padding size, s is the stride, and p_s is the pooling size.

4.1.3 Second CNN architecture (HMB2)

Table 4 and Fig. 6 show a summary of the design of the second CNN architecture. The configuration of the convolutional, max-pooling, batch normalization, and dropout layers is the same as HMB1 architecture. The main difference is the number and size of hidden layers. HMB1 is more complex and has more trainable parameters than HMB2. However, L2 regularization [49] with a value of 10^{-4} is applied to the different convolutional layers. This will help to study the effect and relation between overfitting, regularization, and architecture complexity.

4.1.4 Training parameters

ReLU [50] was used as the activation function for the hidden layers and SoftMax [51] for the output layer. The number of epochs was 50, and the batch size was 32. Different weights initializers [52] were applied: they were LeCun [53–55] normal and uniform initializer, Xavier (Glorot) [56] normal and uniform initializers, and He [57] normal and uniform initializers.

Different optimizers were applied to overcome the overfitting common problem and reach a higher accuracy faster: they were Adam [58, 59] (A Method for Stochastic Optimization), AdaDelta [60] (An Adaptive Learning Rate Method), AdaMax, AdaGrad [61], and stochastic gradient descent (SGD) [62].

The learning rate was 0.01 for AdaGrad and SGD, 0.002 for AdaMax, 0.001 for Adam, and 1.0 for AdaDelta. All code were written in Python programming language with the help of the Python DL library, Keras [63]. The training used Google Colab [64] and a Graphical Processing Unit.

4.1.5 Performance parameters

Performance metrics were accuracy, precision, recall (sensitivity), and F1 values for training and testing phases. **Accuracy** is the ratio of correctly predicted observation to the total observations. It is the most intuitive performance metric; as it shows whether architecture is being correctly trained and how it will perform [65]. Hence, the best architecture is selected according to the highest validation accuracy, from all of the epochs results.

$$\text{Accuracy} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n + \varepsilon} \tag{3}$$

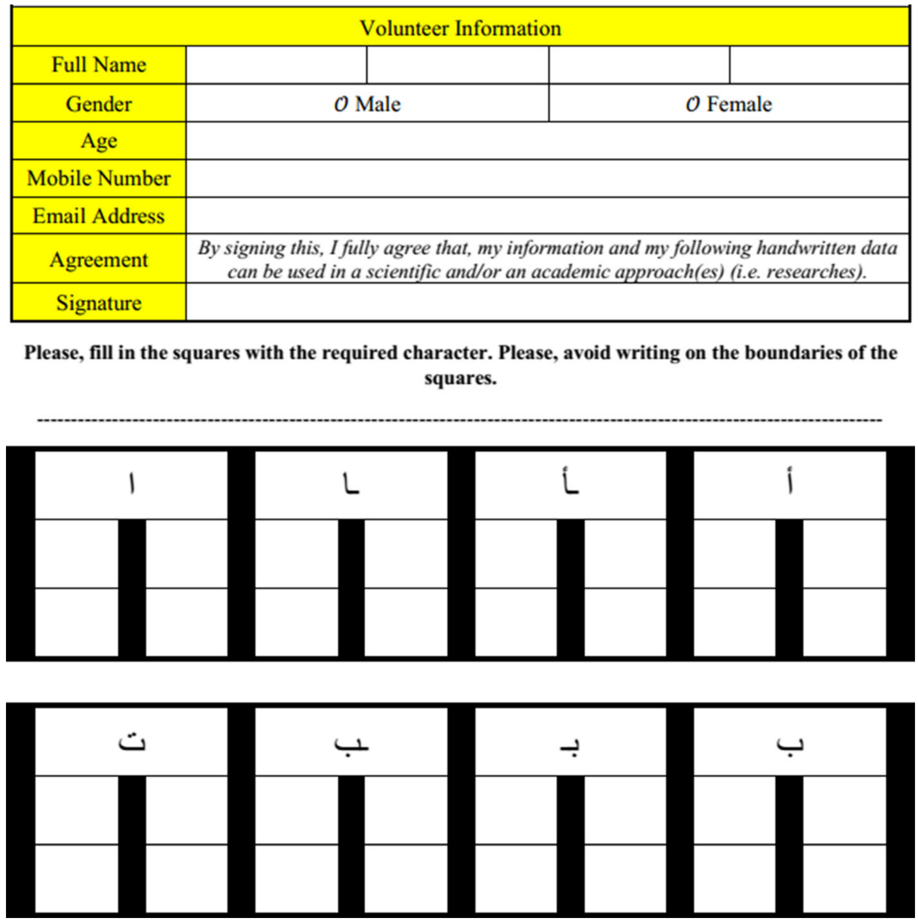
The **precision** is the ratio of correctly predicted positive observations to the total predicted positive observations [66]. It shows how often positive predications occur. The **recall** is the ratio of correctly predicted positive observations to all observations in a specific class. It helps when false negative values are high [67]. The **F1 Score** is the weighted average of precision and recall [68].

$$\text{Precision} = \frac{T_p}{T_p + F_p + \varepsilon} \tag{4}$$

$$\text{Recall} = \frac{T_p}{T_p + F_n + \varepsilon} \tag{5}$$

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

The previous four equations from 3 to 6 [69, 70] demonstrate the different metrics where T_p, T_n, F_p, and F_n are the true positive, true negative, false positive, and false negative values, respectively. Equation 7 demonstrates the loss function. For multi-class classification, categorical



Volunteer Information				
Full Name				
Gender	*O* Male		*O* Female	
Age				
Mobile Number				
Email Address				
Agreement	*By signing this, I fully agree that, my information and my following handwritten data can be used in a scientific and/or an academic approach(es) (i.e. researches).*			
Signature				

Please, fill in the squares with the required character. Please, avoid writing on the boundaries of the squares.

Fig. 7 Sample from the seven pages template (the first page)

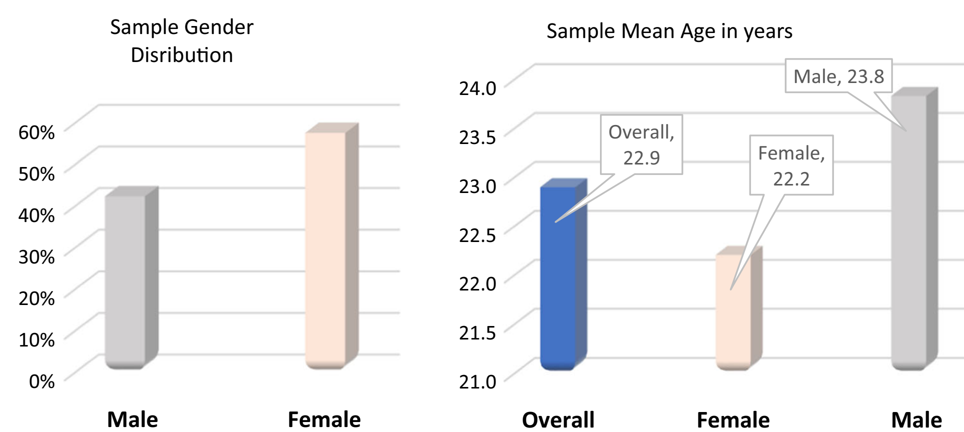


Fig. 8 The age and gender of the sharing persons

cross-entropy (SoftMax loss function) is used. The lower the value, the more favorable [15, 71].

$$\text{SoftMax Loss function} = -\sum_{i=1}^{N} y_i * \log(\hat{y}_i) \tag{7}$$

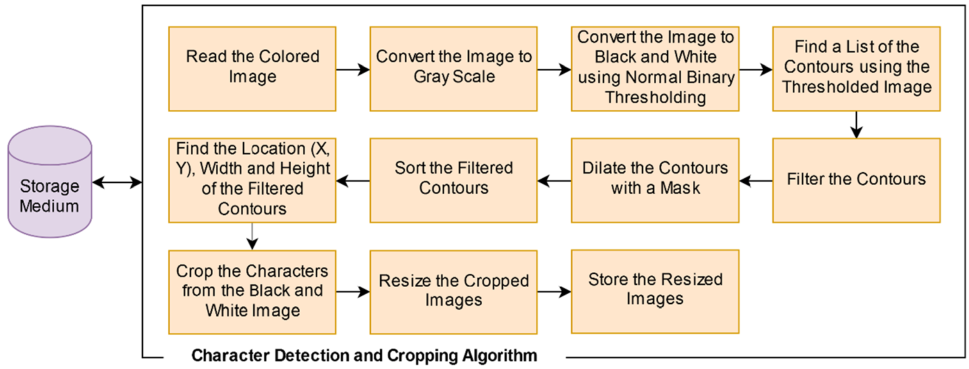
where N is the number of records in the dataset, y_i is the accurate output value, and $\hat{y}_i$ is the architecture predicted value. Epsilon (ε: a very small value) can be added in the denominators to avoid the division by zero.

4.2 Data collection

4.2.1 Tool production

A seven-page dataset template is constructed from the different Arabic characters and numbers; where the characters are drawn so that, the users can mimic them. Taking in consideration, they have different drawings, according to their positions in the word such as: isolated, middle, begin

Fig. 9 Character detection and cropping algorithm



Algorithm 2: Character Detection and Cropping Algorithm

Data: Images dataset
Result: Manipulated characters
parameters initializations;
for *j = 1 to numOfImages* **do**
  $cImg = readColorImage(imagePath)$;
  $gImg = convertGray(cImg)$;
  $bImg = binarizeImage(gImg)$;
  $contours = findContours(bImg)$;
  $filtered = filterContours(contours)$;
  $dilated = dilateContours(filtered)$;
  $sorted = sortContours(dilated)$;
  $locations = locateContours(sorted)$;
  $characters = cropCharacters(locations)$;
  $resized = resizeCharacters(characters)$;
  $storeCharacters(resized)$;
end

Fig. 10 Character detection and cropping algorithm

and end [72]; the constructed template has 116 characters and numerals (digits), and each element has four squares for writing, so, there are 464 squares. One character is duplicated; hence, the total unique elements are 115. (*An example subpage from the seven pages is shown in Fig.* 7).

4.2.2 Sample

Different persons are invited on a voluntary basis to share in this phase of the study. The aim and study design are well-explained to them. 155 volunteers accepted to share. The "seven-page dataset" is printed and distributed to the study sample. The dataset is collected from 125 persons (81% if the invited sample). *The sample age and gender are shown in Fig.* 8. The age has a range of 9 to 82 years,

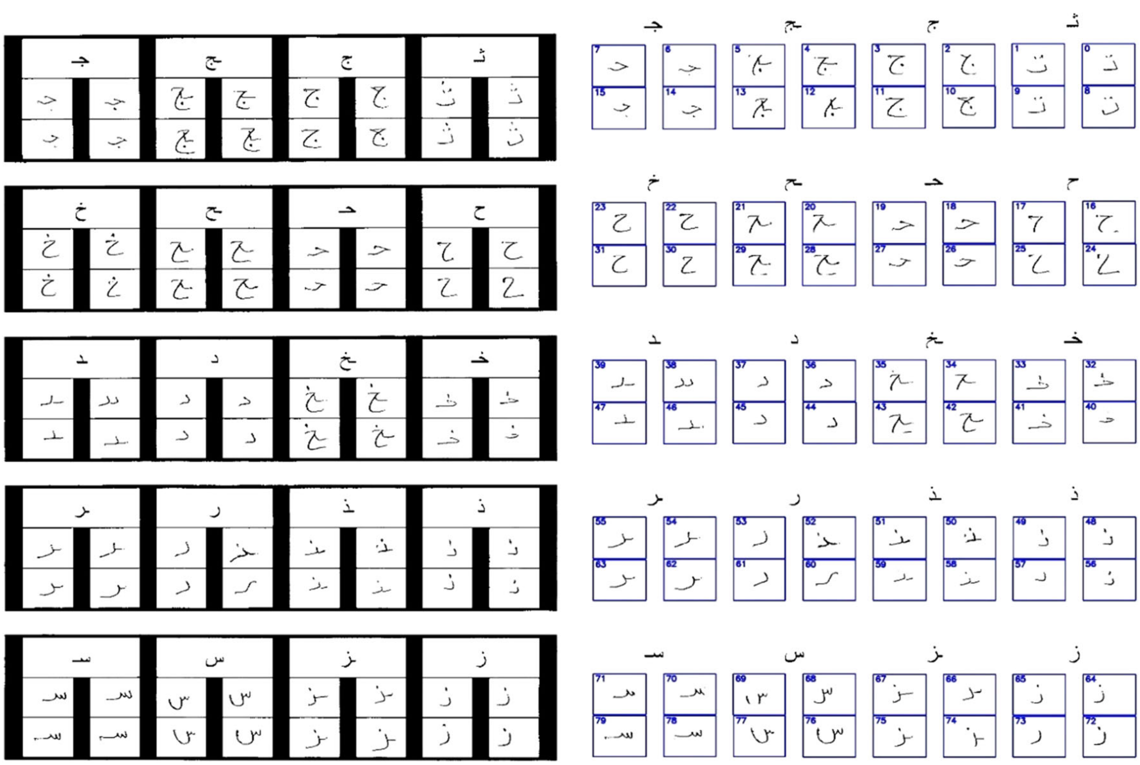


Fig. 11 Left image: one of the input images. Right image: the corresponding output of that image

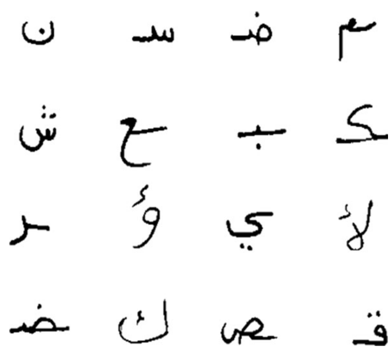


Fig. 12 16 different output characters

and the mean value±Standard Deviation is 22.9 years± 8.8.

4.2.3 Data identification

The pages are collected from the volunteers after completion. Each volunteer has a separate unique identifier on the form "AHCR_xxxxx". The "Xs" are replaced with a unique identifier code starting from 00001.

4.2.4 Data re-coding

The collected pages are scanned using the "HP Scanjet Pro 3000 s2" scanner (company name, and data, take from the back of the scanner). The received file identifiers are re-coded on the form to "AHCR_xxxxx_Dy"; where, the "y" indicates the dataset page number from 1 to 7.

4.3 Dataset production

4.3.1 Character detection

After applying the annotation on the images and collecting them together, each page was passed through a set of layers as shown in the block diagram in Fig. 9.

4.3.2 The algorithm

The algorithm block diagram uses black and white image conversion, finding contours, filtering them, dilating them, sorting them, finding the (x, y) coordinates, width and height of them after sorting, cropping the characters,

Table 5 Generated pickle files with different configurations

Image size	Applied data augmentation?	No. of output images	File name	File size
32×32	No	54,115	HMBD_NO_AUG_32_54115.p	214 MB
64×64	No	54,115	HMBD_NO_AUG_64_54115.p	848 MB
32×32	Yes (1:4)	216,460	HMBD_ AUG_32_216460.p	940 MB
32×32	Yes (1:6)	324,690	HMBD_ AUG_32_324690.p	1.25 GB
128×128	No	54,115	HMBD_NO_AUG_128_54115.p	3.30 GB
32×32	Yes (1:16)	865,840	HMBD_AUG_32_865840.p	3.60 GB

Table 6 Study experiments and their configurations

Experiment	Architecture	Dataset	Size	Augmented?	Image Size
1	HMB1	HMBD	54,115	No	32×32
2	HMB2	HMBD	54,115	No	32×32
3	HMB1	HMBD	54,115	No	64×64
4	HMB2	HMBD	54,115	No	64×64
5	HMB1	HMBD	216,460	Yes	32×32
6	HMB2	HMBD	216,460	Yes	32×32
7	HMB1	HMBD	865,840	Yes	32×32
8	HMB2	HMBD	865,840	Yes	32×32
9	HMB1	CMATER	3000	No	32×32
10	HMB2	CMATER	3000	No	32×32
11	HMB1	CMATER	30,000	Yes	32×32
12	HMB2	CMATER	30,000	Yes	32×32
13	HMB1	AIA9k	8974	No	32×32
14	HMB2	AIA9k	8974	No	32×32
15	HMB1	AIA9k	89,740	Yes	32×32
16	HMB2	AIA9k	89,740	Yes	32×32

resizing them and storing the resized images in the corresponding folder on a storage medium (i.e., hard disk, or cloud storage). The output of the algorithm is the manipulated characters. The algorithm pseudocode for a single input image is presented in Algorithm 2 (Fig. 10).

4.4 Dataset processing and representation

4.4.1 Image processing

Figure 11 shows one of the inputs images before and after processing. Figure 12 shows 16 different characters of the output characters. The total number of collected characters is 58,000 characters. After removing the unwritten, unclear, or bad characters manually, the remaining number is 54,115 characters. The ratio of these filtered images to the total number of unfiltered characters is 93.30%.

4.4.2 Data augmentation

The dataset files are compiled multiple times, with different configurations and stored in Python pickle files to be easily used later in the training process or further working. Images were optionally transformed by 15° rotations, 0.1 width and height shifts, 0.2 shears relative to size and zoom factors of 0.2. These configurations are selected to add only a small change on the images to save it from distortions or to be changed to another letter (i.e., "٦" and "٢"). Table 5 shows the different overall configurations and sizes of output pickle files. The table rows were sorted by the file size ascendingly.

$(1:x)$ in the table means that each input image has another "$x-1$" augmented images, so the overall number of each input image is "x." The filename is initialized by "HMBD" which is the name of the presented dataset. "AUG", or "NO_AUG" follows the dataset name which indicates if a data augmentation is applied or not, respectively. The image size followed by the number of images terminates the filename. The "p" extension indicates that it is a pickle Python file.

Table 8 Cross-validation experiments and their configurations

Phase	Architecture	Dataset	Size	Image size
1	[24]	HMBD	54,114	32×32
2	HMB1	AHCD1 [24]	16,800	32×32
	HMB2	AHCD1 [24]	16,800	32×32

4.5 Validation of the AHCR-DLS with datasets

HMBD, CMATER, and AIA9k datasets were used to validate the AHCR-DLS using the two presented architectures in the current study to validate them and see if the system can work and generalize for different datasets. Table 6 shows the different experiments and their configurations that will be performed. Categorical cross-entropy, ReLU, and SoftMax were the common loss function, hidden activation function, and output activation function, respectively. 64, and 32 were the common number of epochs and batch size, respectively. The used datasets were split into: 95% for training and validation (split internally into 95% for training and 5% for validation) and 5% for testing. Different optimizers (Adam, AdaDelta, SGD, AdaMax, and AdaGrad) and weight initializers (He Normal, He Uniform, Glorot Normal, Glorot Uniform, LeCun Normal and LeCun Uniform) were applied.

4.6 Cross-validation testing

Cross-validation was performed between the presented study architectures and datasets and versus a selected control architecture and a dataset presented in [24] in two phases. The first phase was to test the presented dataset HMBD versus their architecture generalization. The second phase was to test the presented architectures' generalization on their dataset.

Table 7 The design of the CNN architecture in [24]

Layer type	Output shape	Layer type	Output shape
2D convolutional layer	$(28, 28, 80)$	Flatten layer	(1600)
2D max-pooling layer	$(14, 14, 80)$	Dense layer	(1024)
Batch normalization	$(14, 14, 80)$	Dropout layer	(1024)
2D convolutional layer	$(10, 10, 64)$	Dense layer	(512)
2D max-pooling layer	$(5, 5, 64)$	Dropout layer	(1024)
Batch normalization	$(5, 5, 64)$	Dense layer	(115)

Table 9 Comparing HMBD with different AHCR datasets

Database	Data type	No. of tested writers	Dataset size
HMBD	115 (105 Characters and 10 Numbers)	125	54,115
AHCD [24, 28]	28 Characters	700	16,800
AIA9K [27]	28 Characters	107	8737
ADBase & MADBase [73]	10 Numbers	60	70,000

Table 10 Training and testing reported results of Experiment 1

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.04	99.1	0.99	99.2	99.1	0.56	90.1	0.90	90.4	89.8	2301
	AdaDelta	0.05	99.3	0.99	99.3	99.3	0.67	90.5	0.91	90.9	90.4	2480
	SGD	0.05	98.6	0.99	98.7	98.5	0.49	88.6	0.89	89.7	88.1	1838
	AdaMax	0.05	98.6	0.99	98.6	98.5	0.52	88.8	0.89	89.2	88.5	2209
	AdaGrad	0.04	98.9	0.99	99.0	98.9	0.50	88.4	0.89	89.4	87.8	2080
He Uniform	Adam	0.05	98.7	0.99	98.7	98.7	0.52	89.8	0.90	90.4	89.5	2356
	AdaDelta	**0.05**	**99.2**	**0.99**	**99.2**	**99.2**	**0.69**	**90.7**	**0.91**	**91.0**	**90.5**	**2555**
	SGD	0.05	98.9	0.99	98.9	98.8	0.57	87.7	0.88	88.5	87.3	1934
	AdaMax	0.04	99.4	0.99	99.4	99.4	0.59	90.0	0.90	90.2	89.8	2180
	AdaGrad	0.04	99.4	0.99	99.4	99.3	0.51	89.1	0.89	89.7	88.7	2078
Glorot (Xavier) Normal	Adam	0.04	99.1	0.99	99.1	99.1	0.57	90.2	0.90	90.6	90.0	2337
	AdaDelta	0.05	99.0	0.99	99.0	99.0	0.63	90.3	0.90	90.5	90.1	2539
	SGD	0.12	95.9	0.96	96.4	95.5	0.43	88.2	0.89	89.7	87.3	1904
	AdaMax	0.04	99.3	0.99	99.3	99.3	0.58	90.2	0.90	90.5	90.0	2160
	AdaGrad	0.04	99.1	0.99	99.1	99.1	0.47	89.2	0.90	90.1	88.9	2111
Glorot (Xavier) Uniform	Adam	0.04	99.1	0.99	99.2	99.1	0.55	89.6	0.90	89.9	89.4	1738
	AdaDelta	0.05	99.2	0.99	99.2	99.2	0.71	89.9	0.90	90.3	89.7	1907
	SGD	0.03	99.2	0.99	99.2	99.1	0.49	89.1	0.89	89.8	88.8	1458
	AdaMax	0.04	99.2	0.99	99.2	99.2	0.58	89.7	0.90	90.1	89.3	1694
	AdaGrad	0.04	99.4	0.99	99.4	99.3	0.54	89.1	0.89	89.5	88.8	1622
LeCun Normal	Adam	0.04	98.9	0.99	99.0	98.9	0.55	89.5	0.90	89.9	89.4	1815
	AdaDelta	0.04	99.2	0.99	99.2	99.2	0.63	90.1	0.90	90.6	90.0	1999
	SGD	0.04	98.9	0.99	98.9	98.8	0.52	88.8	0.89	89.4	88.5	1523
	AdaMax	0.04	99.1	0.99	99.1	99.1	0.56	89.6	0.90	90.1	89.5	1730
	AdaGrad	0.03	99.4	0.99	99.5	99.4	0.57	89.0	0.89	89.5	88.7	1690
LeCun Uniform	Adam	0.04	99.1	0.99	99.2	99.1	0.53	90.5	0.91	90.9	90.2	1897
	AdaDelta	0.07	97.7	0.98	97.8	97.6	0.54	89.2	0.89	90.0	88.9	2023
	SGD	0.05	98.7	0.99	98.7	98.6	0.48	88.4	0.89	89.6	88.1	1591
	AdaMax	0.04	98.9	0.99	99.0	98.9	0.53	89.9	0.90	90.3	89.5	1773
	AdaGrad	0.04	98.9	0.99	99.0	98.9	0.48	89.1	0.89	89.5	88.6	1708

4.6.1 Phase 1

The architecture in [24] was compiled and tested on HMBD without data augmentation using a dataset pickle file named: "HMBD_NO_AUG_32_54115.p". Their architecture has a design shown in Table 7. The number of classes in the dense output layer was modified from 28 to 115 to match HMBD' classes. They used SoftMax as an output activation function and ReLU as an activation function for the hidden layers. They applied L2

Table 11 Training and testing reported results of Experiment 2

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.34	94.9	0.95	95.6	94.3	0.60	87.8	0.88	89.0	86.8	1143
	AdaDelta	0.26	97.6	0.98	97.8	97.4	0.62	89.2	0.90	90.7	88.4	1211
	SGD	0.10	98.6	0.99	98.7	98.6	0.48	89.2	0.89	89.9	88.9	988
	AdaMax	0.17	98.5	0.99	98.6	98.4	0.59	88.8	0.89	89.8	88.5	1115
	AdaGrad	0.14	97.4	0.97	97.8	97.0	0.45	88.1	0.88	89.6	87.1	1046
He Uniform	Adam	0.29	96.4	0.96	96.8	96.1	0.63	88.2	0.88	89.3	87.1	1249
	AdaDelta	0.26	97.5	0.97	97.7	97.2	0.62	88.4	0.89	90.3	87.3	1341
	SGD	0.11	98.2	0.98	98.3	98.1	0.48	88.7	0.89	89.5	88.1	1074
	AdaMax	0.17	98.2	0.98	98.3	98.1	0.60	88.5	0.89	89.2	88.2	1207
	AdaGrad	0.14	97.2	0.97	97.6	96.8	0.47	87.9	0.88	89.3	86.8	1145
Glorot (Xavier) Normal	Adam	0.27	97.2	0.97	97.4	97.0	0.64	88.2	0.88	89.3	87.5	1283
	AdaDelta	0.28	96.8	0.97	97.1	96.4	0.63	88.1	0.88	89.4	87.2	1384
	SGD	0.11	98.2	0.98	98.3	98.0	0.51	87.9	0.88	88.8	87.3	1098
	AdaMax	0.17	98.3	0.98	98.4	98.3	0.60	88.4	0.89	89.0	88.1	1233
	AdaGrad	0.14	97.4	0.97	97.7	97.1	0.48	87.7	0.88	88.6	87.0	1187
Glorot (Xavier) Uniform	Adam	0.29	96.5	0.97	96.9	96.2	0.63	88.6	0.89	90.0	87.8	1320
	AdaDelta	0.27	97.1	0.97	97.3	96.9	0.63	88.4	0.89	89.8	87.8	1419
	SGD	0.11	98.2	0.98	98.3	98.0	0.50	88.2	0.89	89.3	87.8	1135
	AdaMax	0.17	98.4	0.98	98.5	98.4	0.65	88.1	0.88	88.7	87.7	1276
	AdaGrad	0.13	97.7	0.98	97.9	97.4	0.48	87.7	0.88	89.0	86.9	1227
LeCun Normal	Adam	0.26	97.4	0.97	97.6	97.2	0.63	88.5	0.89	89.6	88.1	1354
	AdaDelta	0.26	97.3	0.97	97.5	97.1	0.65	88.5	0.89	90.0	87.7	1442
	SGD	0.10	98.5	0.99	98.6	98.4	0.51	88.7	0.89	89.6	88.2	1183
	AdaMax	**0.17**	**98.5**	**0.99**	**98.5**	**98.4**	**0.58**	**89.3**	**0.90**	**90.1**	**88.9**	**1304**
	AdaGrad	0.15	97.0	0.97	97.4	96.6	0.48	88.4	0.89	89.7	87.4	1305
LeCun Uniform	Adam	0.29	96.7	0.97	97.0	96.4	0.64	87.8	0.88	89.1	87.0	1299
	AdaDelta	0.31	95.8	0.96	96.5	95.1	0.64	87.3	0.88	89.4	85.7	1383
	SGD	0.10	98.5	0.99	98.6	98.4	0.47	89.1	0.89	90.1	88.7	1074
	AdaMax	0.17	98.4	0.98	98.5	98.3	0.65	88.1	0.88	89.0	87.7	1224
	AdaGrad	0.18	96.4	0.96	97.0	95.9	0.49	87.7	0.88	89.1	86.9	1192

regularization with a value of 0.001 and used the SGD optimizer with a learning rate α equals 0.01 and a momentum γ equals 0.1.

4.6.2 Phase 2

The dataset in [24] was composed of 16,800 handwritten characters written by 60 participants with an age range of 19 to 40 years old and 90% of them were right-handed. The dataset was rescaled to 32×32. This dataset was used with this study described architecture.

Table 8 summarizes the two phases of experiments with their configurations. Categorical cross-entropy, ReLU, and SoftMax were the common loss function, hidden activation function, and output activation function, respectively. 64 and 32 were the common number of epochs and batch size, respectively. The used datasets were split into: 95% for training and validation (split internally into 95% for training and 5% for validation) and 5% for testing. For the second phase only, different optimizers (Adam, AdaDelta, SGD, AdaMax, and AdaGrad) and weight initializers (He Normal, He Uniform, Glorot Normal, Glorot Uniform, LeCun Normal and LeCun Uniform) were applied.

5 Experiments results and discussion

5.1 Comparison of the newly presented dataset with the previous datasets

The presented dataset, HMBD, in the current research captures the different positions of the characters; isolated,

Table 12 Training and testing reported results of Experiment 3

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.03	99.2	0.99	99.2	99.1	0.58	88.7	0.89	89.3	88.4	3518
	AdaDelta	0.04	99.3	0.99	99.4	99.3	0.73	89.4	0.90	90.0	89.4	3851
	SGD	0.03	99.3	0.99	99.4	99.3	0.58	88.9	0.89	89.4	88.4	3027
	AdaMax	**0.03**	**99.4**	**0.99**	**99.5**	**99.4**	**0.64**	**89.8**	**0.90**	**90.1**	**89.5**	**3361**
	AdaGrad	0.03	99.3	0.99	99.4	99.3	0.54	88.1	0.88	88.8	87.6	3160
He Uniform	Adam	0.05	98.6	0.99	98.7	98.6	0.57	88.5	0.89	89.1	88.1	3513
	AdaDelta	0.04	99.2	0.99	99.2	99.2	0.68	89.0	0.89	89.6	88.8	3770
	SGD	0.32	89.7	0.90	91.4	88.2	0.84	79.3	0.80	82.1	77.5	2987
	AdaMax	0.03	99.3	0.99	99.4	99.3	0.69	89.5	0.90	89.8	89.3	3370
	AdaGrad	0.04	99.1	0.99	99.2	99.0	0.50	88.3	0.89	89.2	87.9	3316
Glorot (Xavier) Normal	Adam	0.04	98.8	0.99	98.9	98.8	0.53	89.6	0.90	90.2	89.4	3148
	AdaDelta	0.04	98.9	0.99	99.0	98.9	0.61	89.0	0.89	89.6	88.7	3423
	SGD	0.04	99.0	0.99	99.1	98.9	0.50	88.6	0.89	89.5	88.0	2621
	AdaMax	0.07	97.6	0.98	97.7	97.5	0.43	88.9	0.89	89.8	88.2	2930
	AdaGrad	0.09	97.3	0.97	97.5	97.0	0.41	88.1	0.88	89.6	87.0	2861
Glorot (Xavier) Uniform	Adam	0.04	98.9	0.99	98.9	98.9	0.58	88.3	0.89	88.9	88.1	3531
	AdaDelta	0.04	99.4	0.99	99.4	99.4	0.78	88.8	0.89	89.2	88.5	3787
	SGD	0.05	98.9	0.99	98.9	98.8	0.59	88.6	0.89	89.3	88.2	3031
	AdaMax	0.03	99.5	1.00	99.5	99.5	0.68	89.1	0.89	89.5	89.1	3412
	AdaGrad	0.03	99.4	0.99	99.4	99.4	0.54	89.3	0.89	89.6	88.8	3294
LeCun Normal	Adam	0.03	99.4	0.99	99.4	99.3	0.64	89.0	0.89	89.5	88.7	3614
	AdaDelta	0.04	98.9	0.99	98.9	98.9	0.59	89.2	0.89	89.8	88.9	3890
	SGD	0.04	99.1	0.99	99.1	99.0	0.51	88.9	0.89	89.5	88.6	3056
	AdaMax	0.03	99.4	0.99	99.4	99.4	0.64	88.9	0.89	89.3	88.6	3392
	AdaGrad	0.03	99.2	0.99	99.3	99.2	0.50	88.3	0.89	89.1	87.9	3248
LeCun Uniform	Adam	0.04	99.1	0.99	99.2	99.1	0.63	88.5	0.89	88.9	88.2	3128
	AdaDelta	0.04	99.3	0.99	99.3	99.2	0.73	89.0	0.89	89.3	88.8	11,734
	SGD	0.05	98.3	0.98	98.4	98.3	0.47	88.3	0.89	89.4	87.8	8428
	AdaMax	0.03	99.3	0.99	99.4	99.3	0.63	89.3	0.90	90.0	89.1	9858
	AdaGrad	0.03	99.3	0.99	99.4	99.3	0.55	88.8	0.89	89.4	88.6	3225

beginning, middle, and end; besides, the numbers. The size of the dataset is larger than those available datasets. Table 9 shows a comparison between the HMBD dataset that is presented in this study and the previously prepared datasets by other authors [24, 28]. Most of the currently available datasets deal with the numbers (from 0 to 9), or the 28 characters in the Arabic language (from “أ” to “ي”) 28 ,27 ,24]].

5.2 Experimenting the AHCR-DLS with datasets

Tables 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 and 26 show the corresponding results of the different experiments mentioned in Table 6, respectively. They show the loss, accuracy, F1, precision, and recall values for training and testing phases. The training time is reported in seconds. The best results (concerning the testing accuracy) are in bold.

Table 10 shows that the AdaDelta optimizer and He Uniform weight initializer report the highest testing accuracy in experiment 1. The testing accuracy reached 90.7% with a loss value of 0.69. The average training time is 1974 s. The average training and testing accuracies are 98.9%, and 89.4%, respectively. The average training and testing losses are 0.05, and 0.55, respectively. Figure 13 shows the experiment training and testing accuracies curves with their trendlines. Figure 14 shows the training time curve with its trendline. The *x*-axis refers to the corresponding experiment rows, and the *y*-axis refers to the

Table 13 Training and testing reported results of Experiment 4

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.25	95.6	0.96	96.2	95.0	0.81	83.1	0.84	85.1	82.0	1558
	AdaDelta	0.23	96.7	0.97	97.1	96.5	0.99	83.2	0.84	85.4	82.5	1630
	SGD	0.15	96.8	0.97	97.1	96.5	0.91	82.6	0.83	83.6	81.9	1476
	AdaMax	0.24	94.9	0.95	95.7	94.1	0.67	85.1	0.86	87.0	84.1	1557
	AdaGrad	0.09	98.8	0.99	98.9	98.7	0.65	85.4	0.86	87.0	84.3	1519
He Uniform	Adam	0.23	96.0	0.96	96.3	95.7	0.87	83.9	0.84	85.2	83.3	1623
	AdaDelta	0.24	96.2	0.96	96.7	95.9	0.97	82.3	0.83	84.2	81.6	1689
	SGD	0.24	94.0	0.94	94.6	93.6	1.17	80.3	0.81	81.3	79.8	1500
	AdaMax	0.11	99.0	0.99	99.0	98.9	0.92	85.4	0.86	86.1	84.9	1586
	AdaGrad	0.11	98.5	0.99	98.6	98.4	0.66	85.7	0.86	86.6	85.2	1526
Glorot (Xavier) Normal	Adam	0.21	96.9	0.97	97.1	96.7	0.87	84.1	0.84	84.9	83.5	1667
	AdaDelta	0.24	96.0	0.96	96.4	95.8	1.11	81.2	0.82	82.6	80.7	1589
	SGD	0.16	96.1	0.96	96.5	95.8	0.79	82.9	0.83	84.6	82.2	1377
	AdaMax	0.13	98.7	0.99	98.7	98.6	0.89	84.7	0.85	85.7	84.4	1469
	AdaGrad	0.11	98.5	0.99	98.6	98.4	0.70	85.3	0.86	86.7	84.6	1424
Glorot (Xavier) Uniform	Adam	0.18	97.4	0.97	97.5	97.2	0.83	84.0	0.85	85.3	83.7	1516
	AdaDelta	0.25	95.8	0.96	96.6	95.2	0.77	84.3	0.85	86.5	83.3	1601
	SGD	0.09	99.0	0.99	99.0	98.9	0.84	85.0	0.85	85.7	84.7	1421
	AdaMax	0.15	97.9	0.98	98.0	97.8	0.96	83.7	0.84	84.9	83.4	1510
	AdaGrad	0.13	97.5	0.98	97.9	97.2	0.63	84.7	0.85	86.4	83.4	1464
LeCun Normal	Adam	0.22	96.3	0.96	96.6	96.0	0.81	84.4	0.85	85.7	83.5	1534
	AdaDelta	0.43	91.2	0.91	92.3	90.5	1.09	80.4	0.81	82.3	79.5	1606
	SGD	0.13	97.3	0.97	97.6	96.9	0.73	84.5	0.85	86.1	83.5	1464
	AdaMax	0.15	97.8	0.98	97.9	97.7	0.99	83.4	0.84	84.3	83.2	1565
	AdaGrad	**0.10**	**98.5**	**0.99**	**98.6**	**98.4**	**0.65**	**86.0**	**0.86**	**87.2**	**85.3**	**1522**
LeCun Uniform	Adam	0.17	97.9	0.98	97.9	97.8	0.89	84.0	0.84	84.8	83.5	1622
	AdaDelta	0.31	93.9	0.94	94.8	93.1	0.73	84.8	0.85	87.2	83.2	1697
	SGD	0.30	92.2	0.92	93.6	91.3	0.98	80.2	0.81	82.9	78.6	1397
	AdaMax	0.15	97.9	0.98	98.1	97.7	0.83	84.4	0.85	85.5	83.9	1348
	AdaGrad	0.10	98.5	0.99	98.6	98.4	0.68	84.9	0.85	86.2	84.4	1522

accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 11 shows that the AdaMax optimizer and LeCun Normal weight initializer report the highest testing accuracy in experiment 2. The testing accuracy reached 89.3% with a loss value of 0.58. The average training time is 1228 s. The average training and testing accuracies are 97.5%, and 88.3%, respectively. The average training and testing losses are 0.20, and 0.57, respectively. Figure 15 shows the experiment training and testing accuracies curves with their trendlines. Figure 16 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 12 shows that the AdaMax optimizer and He Normal weight initializer report the highest testing accuracy in experiment 3. The testing accuracy reached 89.8% with a loss value of 0.64. The average training time is 3983 s. The average training and testing accuracies are 98.7%, and 88.5%, respectively. The average training and testing losses are 0.05, and 0.60, respectively. Figure 17 shows the experiment training and testing accuracies curves with their trendlines. Figure 18 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 13 shows that AdGrad optimizer and LeCun Normal weight initializer report the highest testing

Table 14 Training and testing reported results of Experiment 5

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.02	99.5	1.00	99.5	99.5	0.34	93.2	0.93	93.3	93.1	10,224
	AdaDelta	0.03	99.3	0.99	99.3	99.3	0.38	93.4	0.93	93.6	93.3	9893
	SGD	0.03	99.3	0.99	99.3	99.3	0.35	92.0	0.92	92.3	91.8	7564
	AdaMax	0.02	99.6	1.00	99.6	99.6	0.34	93.5	0.94	93.7	93.5	6505
	AdaGrad	0.02	99.6	1.00	99.6	99.5	0.44	92.0	0.92	92.2	91.8	6133
He Uniform	Adam	0.02	99.4	1.00	99.5	99.4	0.30	93.6	0.94	93.8	93.5	6872
	AdaDelta	0.03	99.5	1.00	99.5	99.5	0.42	93.2	0.93	93.4	93.1	7467
	SGD	0.03	99.3	0.99	99.3	99.2	0.37	91.9	0.92	92.2	91.7	5720
	AdaMax	0.02	99.6	1.00	99.6	99.6	0.36	93.4	0.94	93.6	93.3	6536
	AdaGrad	0.02	99.6	1.00	99.6	99.6	0.40	92.4	0.93	92.7	92.3	7476
Glorot (Xavier) Normal	Adam	0.02	99.5	1.00	99.5	99.5	0.33	93.7	0.94	93.9	93.7	8387
	AdaDelta	**0.02**	**99.6**	**1.00**	**99.6**	**99.6**	**0.41**	**93.8**	**0.94**	**94.0**	**93.7**	**7485**
	SGD	0.02	99.4	0.99	99.4	99.4	0.35	92.9	0.93	93.1	92.9	5866
	AdaMax	0.02	99.6	1.00	99.6	99.6	0.36	93.5	0.94	93.6	93.4	6489
	AdaGrad	0.02	99.6	1.00	99.6	99.5	0.44	91.7	0.92	91.8	91.5	6162
Glorot (Xavier) Uniform	Adam	0.02	99.4	0.99	99.4	99.4	0.32	93.2	0.93	93.5	93.2	6937
	AdaDelta	0.03	99.5	1.00	99.5	99.5	0.42	93.3	0.93	93.5	93.2	7435
	SGD	0.02	99.5	1.00	99.5	99.5	0.33	93.2	0.93	93.4	93.1	5625
	AdaMax	0.02	99.5	1.00	99.5	99.5	0.32	93.7	0.94	93.9	93.6	7857
	AdaGrad	0.02	99.5	1.00	99.5	99.5	0.40	92.1	0.92	92.3	91.9	7556
LeCun Normal	Adam	0.02	99.5	1.00	99.5	99.5	0.31	93.7	0.94	93.9	93.6	6976
	AdaDelta	0.03	99.5	1.00	99.5	99.5	0.42	93.5	0.94	93.7	93.4	7755
	SGD	0.02	99.4	1.00	99.5	99.4	0.35	92.7	0.93	93.0	92.6	5757
	AdaMax	0.02	99.6	1.00	99.6	99.6	0.35	93.4	0.93	93.6	93.3	6431
	AdaGrad	0.02	99.5	1.00	99.6	99.5	0.41	92.0	0.92	92.2	91.8	6045
LeCun Uniform	Adam	0.02	99.5	1.00	99.5	99.5	0.32	93.6	0.94	93.7	93.5	6753
	AdaDelta	0.02	99.5	1.00	99.6	99.5	0.42	93.3	0.93	93.5	93.2	7360
	SGD	0.03	99.3	0.99	99.3	99.3	0.34	93.0	0.93	93.2	92.9	5701
	AdaMax	0.02	99.6	1.00	99.6	99.6	0.34	93.7	0.94	93.8	93.6	6578
	AdaGrad	0.02	99.5	1.00	99.6	99.5	0.41	92.0	0.92	92.2	91.9	6281

accuracy in experiment 4. The testing accuracy reached 86.0% with a loss value of 0.65. The average training time is 1533 s. The average training and testing accuracies are 96.7%, and 83.8%, respectively. The average training and testing losses are 0.19, and 0.85, respectively. Figure 19 shows the experiment training and testing accuracies curves with their trendlines. Figure 20 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 14 shows that the AdaDelta optimizer and Glorot Normal weight initializer report the highest testing accuracy in experiment 5. The testing accuracy reached 93.8% with a loss value of 0.41. The average training time is 6449 s. The average training and testing accuracies are 99.5%, and 93.0%, respectively. The average training and testing losses are 0.02, and 0.37, respectively. Figure 21 shows the experiment training and testing accuracies curves with their trendlines. Figure 22 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 15 shows that the SGD optimizer and LeCun Uniform weight initializer report the highest testing accuracy in experiment 6. The testing accuracy reached 92.2% with a loss value of 0.30. The average training time is 3996 s. The average training and testing accuracies are 94.8%, and 90.7%, respectively. The average training and

Table 15 Training and testing reported results of Experiment 6

Weight Initializer	Optimizer	Training Phase					Testing Phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.36	93.1	0.93	94.1	92.2	0.47	89.9	0.90	91.3	88.7	3842
	AdaDelta	0.43	92.6	0.93	93.6	91.4	0.55	89.6	0.90	91.2	88.2	4167
	SGD	0.14	97.3	0.97	97.6	97.1	0.32	91.7	0.92	92.4	91.1	3276
	AdaMax	0.24	95.8	0.96	96.2	95.4	0.37	91.6	0.92	92.3	91.0	3867
	AdaGrad	0.18	95.1	0.95	95.8	94.4	0.31	90.6	0.91	91.9	89.5	3632
He Uniform	Adam	0.35	93.6	0.94	94.4	92.8	0.46	90.2	0.90	91.4	89.1	4147
	AdaDelta	0.43	92.6	0.93	93.8	91.2	0.55	89.5	0.90	91.3	87.9	4083
	SGD	0.15	96.7	0.97	97.0	96.4	0.31	91.4	0.92	92.2	90.8	3295
	AdaMax	0.24	96.0	0.96	96.4	95.7	0.38	91.6	0.92	92.6	91.0	3732
	AdaGrad	0.17	95.6	0.96	96.2	94.9	0.32	90.7	0.91	91.9	89.7	3491
Glorot (Xavier) Normal	Adam	0.36	93.3	0.93	94.3	92.3	0.47	89.8	0.90	91.2	88.5	3915
	AdaDelta	0.42	92.7	0.93	93.8	91.6	0.54	89.9	0.90	91.6	88.5	4172
	SGD	0.15	96.7	0.97	97.1	96.4	0.31	91.4	0.91	92.2	90.6	3234
	AdaMax	0.24	95.8	0.96	96.2	95.4	0.38	91.6	0.92	92.4	90.9	3508
	AdaGrad	0.16	96.0	0.96	96.5	95.5	0.30	91.2	0.91	92.2	90.3	3343
Glorot (Xavier) Uniform	Adam	0.33	93.9	0.94	94.6	93.1	0.45	90.8	0.91	91.8	89.8	3726
	AdaDelta	0.43	92.3	0.92	93.5	91.1	0.55	89.4	0.89	91.0	88.0	4236
	SGD	0.14	97.0	0.97	97.3	96.7	0.32	91.6	0.92	92.6	91.0	3235
	AdaMax	0.25	95.5	0.95	96.0	95.0	0.38	91.3	0.91	92.3	90.5	4914
	AdaGrad	0.42	92.6	0.93	93.8	91.3	0.54	89.4	0.89	91.0	87.7	4050
LeCun Normal	Adam	0.36	93.3	0.93	94.2	92.4	0.47	90.1	0.90	91.3	89.0	4026
	AdaDelta	0.42	92.6	0.93	93.8	91.3	0.53	89.8	0.90	91.3	88.3	4316
	SGD	0.15	96.8	0.97	97.1	96.5	0.33	91.5	0.92	92.3	90.9	3380
	AdaMax	0.25	95.5	0.96	95.9	95.1	0.39	91.5	0.92	92.4	90.9	3829
	AdaGrad	0.18	95.4	0.95	96.1	94.8	0.31	91.0	0.91	92.1	89.8	4435
LeCun Uniform	Adam	0.34	93.7	0.94	94.5	92.9	0.47	89.8	0.90	91.1	88.8	4976
	AdaDelta	0.43	92.4	0.92	93.8	91.1	0.54	89.6	0.90	91.3	87.8	5374
	SGD	**0.13**	**97.7**	**0.98**	**97.9**	**97.5**	**0.30**	**92.2**	**0.92**	**92.8**	**91.7**	**4266**
	AdaMax	0.24	95.8	0.96	96.2	95.4	0.38	91.1	0.91	91.9	90.5	4824
	AdaGrad	0.14	96.7	0.97	97.2	96.3	0.30	91.3	0.91	92.4	90.4	4596

testing losses are 0.27, and 0.41, respectively. Figure 23 shows the experiment training and testing accuracies curves with their trendlines. Figure 24 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 16 shows that Adam optimizer and LeCun Uniform weight initializer report the highest testing accuracy in experiment 7. The testing accuracy reached 98.4% with a loss value of 0.07. The average training time is 28,595 s. The average training and testing accuracies are 99.8%, and 97.8%, respectively. The average training and testing losses are 0.01, and 0.10, respectively. Figure 25 shows the experiment training and testing accuracies curves with their trendlines. Figure 26 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 17 shows that SGD and He Uniform weight initializer report the highest testing accuracy in experiment 8. The testing accuracy reached 95.4% with a loss value of 0.18. The average training time is 16,245 s. The average training and testing accuracies are 94.3%, and 93.2%, respectively. The average training and testing losses are 0.27, and 0.30, respectively. Figure 27 shows the experiment training and testing accuracies curves with their trendlines. Figure 28 shows the training time curve with its trendline. The x-axis refers to the corresponding

Table 16 Training and testing reported results of Experiment 7

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.01	99.8	1.00	99.8	99.8	0.07	98.2	0.98	98.3	98.2	35,194
	AdaDelta	0.01	99.7	1.00	99.8	99.7	0.10	98.0	0.98	98.1	98.0	31,005
	SGD	0.01	99.7	1.00	99.7	99.7	0.09	97.7	0.98	97.8	97.7	28,300
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.08	98.3	0.98	98.3	98.3	25,521
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.16	96.6	0.97	96.6	96.6	24,206
He Uniform	Adam	0.01	99.8	1.00	99.8	99.8	0.07	98.2	0.98	98.3	98.2	27,081
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.09	98.1	0.98	98.2	98.1	30,401
	SGD	0.01	99.8	1.00	99.8	99.8	0.09	97.8	0.98	97.8	97.7	23,078
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.07	98.2	0.98	98.2	98.1	34,270
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.13	96.9	0.97	97.0	96.9	25,048
Glorot (Xavier) Normal	Adam	0.01	99.8	1.00	99.8	99.8	0.07	98.3	0.98	98.3	98.3	28,252
	AdaDelta	0.01	99.7	1.00	99.7	99.7	0.09	98.1	0.98	98.1	98.1	26,334
	SGD	0.01	99.8	1.00	99.8	99.8	0.10	97.7	0.98	97.8	97.7	22849
	AdaMax	0.01	99.9	1.00	99.9	99.9	0.07	98.3	0.98	98.3	98.3	26,581
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.15	96.7	0.97	96.8	96.7	24,868
Glorot (Xavier) Uniform	Adam	0.01	99.8	1.00	99.8	99.8	0.07	98.1	0.98	98.2	98.1	25,827
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.10	98.1	0.98	98.2	98.1	30,129
	SGD	0.01	99.8	1.00	99.8	99.8	0.09	97.8	0.98	97.9	97.8	29,349
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.08	98.3	0.98	98.3	98.2	25,901
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.14	96.8	0.97	96.8	96.7	31,701
LeCun Normal	Adam	0.01	99.8	1.00	99.8	99.8	0.08	98.2	0.98	98.2	98.1	33,002
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.11	98.0	0.98	98.1	98.0	29,398
	SGD	0.01	99.8	1.00	99.8	99.8	0.10	97.7	0.98	97.8	97.7	23,725
	AdaMax	0.01	99.9	1.00	99.9	99.9	0.07	98.3	0.98	98.3	98.3	33,528
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.16	96.6	0.97	96.7	96.6	32,193
LeCun Uniform	**Adam**	**0.01**	**99.9**	**1.00**	**99.9**	**99.9**	**0.07**	**98.4**	**0.98**	**98.4**	**98.4**	**34,729**
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.09	98.1	0.98	98.1	98.1	28,113
	SGD	0.01	99.8	1.00	99.8	99.8	0.10	97.7	0.98	97.7	97.7	28,772
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.08	98.2	0.98	98.3	98.2	34,266
	AdaGrad	0.01	99.7	1.00	99.7	99.7	0.12	97.1	0.97	97.2	97.0	24,234

experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 18 shows that two scenarios in experiment 9 reach 100% testing accuracies, but AdaDelta optimizer and Glorot (Xavier) Uniform weight initializer report the highest training and testing accuracies. The testing accuracy reached 100% with a loss value of 0. The average training time is 141 s. The average training and testing accuracies are 99.6%, and 97.9%, respectively. The average training and testing losses are 0.02, and 0.07, respectively. Figure 29 shows the experiment training and testing accuracies curves with their trendlines. Figure 30 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 19 shows that Adam optimizer and LeCun Uniform weight initializer report the highest testing accuracy in experiment 10. The testing accuracy reached 100% with a loss value of 0.05. The average training time is 142 s. The average training and testing accuracies are 99.7%, and 98.3%, respectively. The average training and testing losses are 0.06, and 0.09, respectively. Figure 31 shows the experiment training and testing accuracies curves with their trendlines. Figure 32 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 17 Training and testing reported results of Experiment 8

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.40	91.7	0.92	92.7	90.8	0.42	90.8	0.91	92.0	89.9	20,889
	AdaDelta	0.47	91.2	0.91	92.7	89.5	0.49	90.7	0.91	92.2	88.9	20,611
	SGD	0.15	96.9	0.97	97.1	96.6	0.18	95.4	0.96	95.8	95.1	17,437
	AdaMax	0.28	94.1	0.94	94.8	93.5	0.31	93.1	0.93	94.0	92.4	19,322
	AdaGrad	0.12	96.8	0.97	97.1	96.5	0.16	95.0	0.95	95.5	94.6	14,118
He Uniform	Adam	0.39	91.6	0.92	92.9	90.5	0.42	90.9	0.91	92.3	89.7	16,365
	AdaDelta	0.48	90.6	0.91	92.4	88.7	0.51	89.9	0.90	91.9	88.1	17,909
	SGD	**0.14**	**96.9**	**0.97**	**97.1**	**96.7**	**0.18**	**95.4**	**0.96**	**95.8**	**95.1**	**13,268**
	AdaMax	0.28	94.1	0.94	94.7	93.6	0.31	93.0	0.93	93.7	92.5	19,555
	AdaGrad	0.12	96.9	0.97	97.2	96.6	0.16	95.2	0.95	95.7	94.9	13,702
Glorot (Xavier) Normal	Adam	0.39	91.8	0.92	93.1	90.6	0.41	91.1	0.91	92.5	89.8	15,415
	AdaDelta	0.47	91.1	0.91	92.4	89.8	0.49	90.6	0.91	92.0	89.3	16,169
	SGD	0.14	96.8	0.97	97.0	96.6	0.19	95.3	0.95	95.7	95.0	16,564
	AdaMax	0.27	94.6	0.95	95.2	94.0	0.30	93.7	0.94	94.3	93.0	19,878
	AdaGrad	0.12	96.8	0.97	97.2	96.5	0.16	95.1	0.95	95.6	94.7	18,325
Glorot (Xavier) Uniform	Adam	0.48	90.9	0.91	92.4	89.5	0.50	90.3	0.90	91.7	88.9	16,485
	AdaDelta	0.14	96.9	0.97	97.2	96.7	0.18	95.4	0.95	95.8	95.1	13,203
	SGD	0.27	94.3	0.94	94.9	93.7	0.30	93.3	0.93	94.1	92.6	19,922
	AdaMax	0.13	96.5	0.97	96.8	96.2	0.17	94.9	0.95	95.4	94.5	13,465
	AdaGrad	0.12	96.9	0.97	97.2	96.6	0.16	95.1	0.95	95.6	94.7	13,929
LeCun Normal	Adam	0.38	91.9	0.92	93.0	91.0	0.41	91.2	0.91	92.2	90.3	15,449
	AdaDelta	0.47	91.1	0.91	92.5	89.8	0.49	90.6	0.91	92.2	89.3	17,164
	SGD	0.17	96.1	0.96	96.5	95.8	0.21	94.8	0.95	95.3	94.4	12,632
	AdaMax	0.28	94.2	0.94	94.8	93.5	0.31	93.2	0.93	94.0	92.6	14,252
	AdaGrad	0.12	96.8	0.97	97.1	96.5	0.16	95.1	0.95	95.5	94.6	17,546
LeCun Uniform	Adam	0.38	92.2	0.92	93.2	91.2	0.40	91.4	0.92	92.6	90.4	15,879
	AdaDelta	0.47	91.0	0.91	92.6	89.4	0.49	90.5	0.90	92.2	88.8	17,055
	SGD	0.15	96.8	0.97	97.1	96.5	0.19	95.4	0.95	95.8	95.0	13,181
	AdaMax	0.27	94.3	0.94	95.0	93.7	0.31	93.3	0.93	94.1	92.7	14,094
	AdaGrad	0.12	96.9	0.97	97.2	96.6	0.17	95.1	0.95	95.5	94.7	13,569

Table 20 shows that the AdaDelta optimizer and Glorot (Xavier) Uniform weight initializer report the highest testing accuracy in experiment 11. The testing accuracy reached 99.4% with a loss value of 0.05. The average training time is 1000 s. The average training and testing accuracies are 99.8%, and 98.9%, respectively. The average training and testing losses are 0.01, and 0.05, respectively. Figure 33 shows the experiment training and testing accuracies curves with their trendlines. Figure 34 shows the training time curve with its trendline. The *x*-axis refers to the corresponding experiment rows, and the *y*-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 21 shows that the AdaMax optimizer and LeCun Uniform weight initializer report the highest testing accuracy in experiment 12. The testing accuracy reached 99.2% with a loss value of 0.06. The average training time is 751 s. The average training and testing accuracies are 99.8%, and 98.8%, respectively. The average training and testing losses are 0.05, and 0.08, respectively. Figure 35 shows the experiment training and testing accuracies curves with their trendlines. Figure 36 shows the training time curve with its trendline. The *x*-axis refers to the corresponding experiment rows, and the *y*-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

From experiments (9 to 12), accuracy, precision, and recall values are above (or equal) 99.2%, loss values are below (or equal) 0.06 and F1 values are above (or equal) 0.99. HMB1 reports better results than HMB2. Table 22

Table 18 Training and testing reported results of Experiment 9

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.03	99.2	0.99	99.3	99.1	0.14	96.7	0.97	97.3	96.7	113
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.04	99.3	0.99	99.3	98.7	117
	SGD	0.01	99.9	1.00	99.9	99.9	0.03	98.7	0.99	99.3	98.7	94
	AdaMax	0.01	99.9	1.00	99.9	99.9	0.07	98.0	0.98	98.0	98.0	109
	AdaGrad	0.01	99.9	1.00	99.9	99.8	0.07	97.3	0.97	97.3	96.7	104
He Uniform	Adam	0.02	99.6	1.00	99.6	99.6	0.14	95.3	0.95	95.3	95.3	117
	AdaDelta	0.05	98.7	0.99	98.9	98.6	0.06	97.3	0.98	98.0	97.3	128
	SGD	0.02	99.7	1.00	99.7	99.6	0.07	97.3	0.97	97.3	97.3	103
	AdaMax	0.02	99.7	1.00	99.7	99.7	0.05	98.7	0.98	98.7	98.0	119
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.09	97.3	0.97	97.3	97.3	118
Glorot (Xavier) Normal	Adam	0.01	99.5	1.00	99.5	99.5	0.02	99.3	1.00	100.0	99.3	131
	AdaDelta	0.01	99.7	1.00	99.7	99.7	0.01	100.0	1.00	100.0	100.0	141
	SGD	0.00	99.9	1.00	99.9	99.9	0.03	99.3	0.99	99.3	98.7	124
	AdaMax	0.03	98.9	0.99	99.1	98.9	0.09	96.7	0.97	96.7	96.7	135
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.13	95.3	0.96	96.0	95.3	131
Glorot (Xavier) Uniform	Adam	0.01	99.6	1.00	99.6	99.6	0.02	98.7	0.99	98.7	98.7	149
	AdaDelta	**0.01**	**99.8**	**1.00**	**99.8**	**99.8**	**0.00**	**100.0**	**1.00**	**100.0**	**100.0**	**159**
	SGD	0.02	99.6	1.00	99.6	99.4	0.05	98.7	0.99	99.3	98.0	133
	AdaMax	0.06	98.6	0.99	98.8	98.3	0.16	95.3	0.95	95.3	95.3	150
	AdaGrad	0.02	99.7	1.00	99.7	99.7	0.08	96.0	0.96	96.7	96.0	146
LeCun Normal	Adam	0.01	99.8	1.00	99.8	99.8	0.10	98.7	0.99	98.7	98.7	164
	AdaDelta	0.01	99.7	1.00	99.7	99.7	0.15	96.7	0.97	97.3	96.7	170
	SGD	0.01	99.7	1.00	99.7	99.7	0.03	99.3	0.99	99.3	98.7	152
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.03	98.0	0.98	98.6	98.0	170
	AdaGrad	0.03	99.3	0.99	99.4	99.3	0.04	98.7	0.99	98.7	98.7	160
LeCun Uniform	Adam	0.01	99.8	1.00	99.8	99.8	0.03	98.7	0.99	98.7	98.7	179
	AdaDelta	0.00	99.9	1.00	99.9	99.9	0.07	98.7	0.99	98.7	98.7	185
	SGD	0.01	99.8	1.00	99.8	99.8	0.03	98.7	0.98	98.6	98.0	164
	AdaMax	0.01	99.8	1.00	99.9	99.8	0.03	98.0	0.98	98.0	98.0	181
	AdaGrad	0.03	99.3	0.99	99.3	99.2	0.13	96.0	0.96	96.7	95.3	175

shows a comparison between the study reported accuracies, and other authors (researches) reported accuracies:

Figure 37 shows the confusion matrices (top left, top right, bottom left, and bottom right) of the last four experiments (9 to 12), respectively.

Table 23 shows that the AdaDelta optimizer and He Uniform weight initializer report the highest testing accuracy in experiment 13. The testing accuracy reached 92.7% with a loss value of 0.42. The average training time is 334 s. The average training and testing accuracies are 92.6%, and 86.6%, respectively. The average training and testing losses are 0.24, and 0.48, respectively. Figure 38 shows the experiment training and testing accuracies curves with their trendlines. Figure 39 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 24 shows that two scenarios in experiment 14 reach 93.3% testing accuracies, but AdaGrad optimizer and He Uniform weight initializer report the highest training and testing accuracies. The testing accuracy reached 93.3% with a loss value of 0.38. The average training time is 179 s. The average training and testing accuracies are 99.4%, and 91.9%, respectively. The average training and testing losses are 0.09, and 0.47, respectively. Figure 40 shows the experiment training and testing accuracies curves with their trendlines. Figure 41 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a

Table 19 Training and testing reported results of Experiment 10

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.09	99.3	0.99	99.3	99.3	0.12	98.7	0.99	98.7	98.7	124
	AdaDelta	0.05	99.8	1.00	99.8	99.8	0.08	98.7	0.99	98.7	98.7	129
	SGD	0.05	99.8	1.00	99.8	99.8	0.08	98.7	0.99	99.3	98.7	113
	AdaMax	0.05	99.8	1.00	99.8	99.8	0.10	96.7	0.97	96.7	96.7	125
	AdaGrad	0.06	99.8	1.00	99.8	99.8	0.10	98.0	0.98	98.0	98.0	118
He Uniform	Adam	0.07	99.8	1.00	99.8	99.8	0.12	98.0	0.98	98.0	98.0	133
	AdaDelta	0.06	99.7	1.00	99.7	99.7	0.08	99.3	0.99	99.3	99.3	135
	SGD	0.06	99.7	1.00	99.7	99.7	0.12	98.0	0.98	98.0	98.0	121
	AdaMax	0.05	99.9	1.00	99.9	99.8	0.08	99.3	0.99	99.3	99.3	133
	AdaGrad	0.06	99.8	1.00	99.8	99.8	0.10	98.7	0.99	98.7	98.7	127
Glorot (Xavier) Normal	Adam	0.06	99.8	1.00	99.8	99.8	0.12	98.7	0.99	98.7	98.7	141
	AdaDelta	0.05	99.8	1.00	99.8	99.8	0.11	98.0	0.98	98.0	98.0	146
	SGD	0.05	99.8	1.00	99.8	99.8	0.07	97.3	0.97	97.3	97.3	128
	AdaMax	0.07	99.5	1.00	99.5	99.5	0.15	94.7	0.95	94.7	94.7	141
	AdaGrad	0.05	99.8	1.00	99.8	99.8	0.04	99.3	0.99	99.3	99.3	134
Glorot (Xavier) Uniform	Adam	0.07	99.8	1.00	99.8	99.8	0.13	98.7	0.99	98.7	98.7	150
	AdaDelta	0.10	98.4	0.98	98.5	98.3	0.10	97.3	0.98	98.0	97.3	151
	SGD	0.07	99.3	0.99	99.4	99.3	0.08	98.0	0.98	98.7	98.0	136
	AdaMax	0.05	99.8	1.00	99.8	99.8	0.11	96.7	0.97	96.7	96.7	149
	AdaGrad	0.07	99.5	1.00	99.5	99.5	0.07	99.3	0.99	99.3	99.3	141
LeCun Normal	Adam	0.09	99.4	0.99	99.4	99.4	0.15	97.3	0.97	97.3	97.3	157
	AdaDelta	0.07	99.5	1.00	99.5	99.5	0.06	99.3	0.99	99.3	99.3	160
	SGD	0.05	99.8	1.00	99.8	99.8	0.08	99.3	0.99	99.3	99.3	145
	AdaMax	0.05	99.9	1.00	99.9	99.9	0.06	99.3	0.99	99.3	99.3	158
	AdaGrad	0.07	99.6	1.00	99.7	99.6	0.10	97.3	0.97	97.3	97.3	148
LeCun Uniform	**Adam**	**0.06**	**99.7**	**1.00**	**99.7**	**99.7**	**0.05**	**100.0**	**1.00**	**100.0**	**100.0**	**166**
	AdaDelta	0.04	99.9	1.00	99.9	99.9	0.08	99.3	0.99	99.3	99.3	166
	SGD	0.05	99.8	1.00	99.8	99.8	0.10	98.0	0.98	98.0	98.0	154
	AdaMax	0.06	99.6	1.00	99.6	99.6	0.08	99.3	0.99	99.3	99.3	165
	AdaGrad	0.05	99.8	1.00	99.8	99.8	0.07	98.0	0.98	98.7	98.0	155

percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 25 shows that the AdaMax optimizer and He Uniform weight initializer report the highest testing accuracy in experiment 15. The testing accuracy reached 99.0% with a loss value of 0.06. The average training time is 3116 s. The average training and testing accuracies are 99.9%, and 98.6%, respectively. The average training and testing losses are 0.01, and 0.08, respectively. Figure 42 shows the experiment training and testing accuracies curves with their trendlines. Figure 43 shows the training time curve with its trendline. The *x*-axis refers to the corresponding experiment rows, and the *y*-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 26 shows that the SGD optimizer and He Normal weight initializer report the highest testing accuracy in experiment 16. The testing accuracy reached 98.4% with a loss value of 0.10. The average training time is 1764 s. The average training and testing accuracies are 99.4%, and 97.8%, respectively. The average training and testing losses are 0.08, and 0.13, respectively. Figure 44 shows the experiment training and testing accuracies curves with their trendlines. Figure 45 shows the training time curve with its trendline. The *x*-axis refers to the corresponding experiment rows, and the *y*-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

From this finding, it can be concluded that the two newly presented architectures in the current study can be

Table 20 Training and testing reported results of Experiment 11

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.01	99.9	1.00	99.9	99.9	0.03	99.3	0.99	99.3	99.3	959
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.05	99.2	0.99	99.2	99.2	1027
	SGD	0.02	99.5	1.00	99.5	99.4	0.05	98.7	0.99	98.7	98.6	787
	AdaMax	0.00	99.9	1.00	99.9	99.9	0.06	98.8	0.99	98.8	98.8	910
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.06	98.7	0.99	98.7	98.6	863
He Uniform	Adam	0.01	99.9	1.00	99.9	99.9	0.05	98.9	0.99	98.9	98.9	976
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.04	99.1	0.99	99.1	99.1	1058
	SGD	0.01	99.6	1.00	99.6	99.6	0.04	98.8	0.99	98.8	98.7	824
	AdaMax	0.01	99.9	1.00	99.9	99.9	0.04	99.3	0.99	99.3	99.3	937
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.04	98.8	0.99	98.8	98.7	902
Glorot (Xavier) Normal	Adam	0.01	99.9	1.00	99.9	99.9	0.06	98.7	0.99	98.7	98.7	1016
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.06	99.3	0.99	99.3	99.3	1093
	SGD	0.02	99.5	1.00	99.5	99.5	0.05	98.7	0.99	98.8	98.7	857
	AdaMax	0.01	99.9	1.00	99.9	99.9	0.05	98.8	0.99	98.9	98.8	980
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.03	99.3	0.99	99.3	99.3	937
Glorot (Xavier) Uniform	Adam	0.01	99.9	1.00	99.9	99.9	0.05	98.9	0.99	98.9	98.9	1062
	AdaDelta	**0.00**	**99.9**	**1.00**	**99.9**	**99.9**	**0.05**	**99.4**	**0.99**	**99.4**	**99.4**	**1143**
	SGD	0.02	99.6	1.00	99.6	99.6	0.05	98.6	0.99	98.6	98.6	907
	AdaMax	0.01	99.9	1.00	99.9	99.9	0.04	98.8	0.99	98.9	98.8	1028
	AdaGrad	0.01	99.6	1.00	99.7	99.6	0.05	98.6	0.99	98.6	98.6	983
LeCun Normal	Adam	0.00	99.9	1.00	99.9	99.9	0.03	99.1	0.99	99.1	99.1	1100
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.06	98.8	0.99	98.8	98.8	1173
	SGD	0.02	99.4	0.99	99.4	99.3	0.05	98.4	0.98	98.5	98.4	933
	AdaMax	0.01	99.9	1.00	99.9	99.9	0.04	99.2	0.99	99.2	99.2	1061
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.05	98.6	0.99	98.6	98.6	1017
LeCun Uniform	Adam	0.01	99.8	1.00	99.8	99.8	0.05	98.8	0.99	98.8	98.8	1133
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.05	99.1	0.99	99.1	99.1	1227
	SGD	0.02	99.6	1.00	99.6	99.5	0.04	98.8	0.99	98.8	98.8	965
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.05	98.6	0.99	98.6	98.6	1097
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.05	98.9	0.99	98.9	98.9	1058

used for different datasets, with a high-quality output. Besides that, they produce higher accuracies than two reported and published accuracies. As mentioned, the differences between the two described architectures are in the design of the hidden layers, such as the number and type of layers, the number and the size of kernels, and the size of strides. This concludes also two main things (from the hardware point of view): (i) HMB1 can achieve higher accuracy than HMB2 and can be used with devices with high-level hardware such as Gaming devices. (ii) HMB2 can achieve a bit lower accuracy than HMB2 and can be used with devices such as mobile phones (they contain low-level hardware compared with Gaming devices).

5.3 Cross-validation testing

5.3.1 Phase 1: (test HMBD versus the selected control architecture)

After training their architecture, the results are presented in Table 27.

5.3.2 Comments on the first phase

The training and testing accuracies reach max 51.0% and 47.7%, respectively. The losses are higher than 1.8, and the F1 scores are below 0.5. These values proved that their control architecture cannot generalize for the current study presented dataset. This may be due to the inability of the

Table 21 Training and testing reported results of Experiment 12

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.07	99.8	1.00	99.8	99.8	0.10	98.9	0.99	98.9	98.9	728
	AdaDelta	0.05	99.8	1.00	99.8	99.8	0.09	98.5	0.99	98.5	98.5	789
	SGD	0.04	99.9	1.00	99.9	99.9	0.09	98.5	0.99	98.5	98.5	612
	AdaMax	0.03	99.9	1.00	99.9	99.9	0.09	98.8	0.99	98.8	98.8	698
	AdaGrad	0.04	99.9	1.00	99.9	99.9	0.07	98.9	0.99	98.9	98.9	664
He Uniform	Adam	0.08	99.5	1.00	99.6	99.5	0.14	98.1	0.98	98.1	98.1	752
	AdaDelta	0.05	99.8	1.00	99.8	99.8	0.08	99.1	0.99	99.1	99.1	825
	SGD	0.04	99.9	1.00	99.9	99.9	0.08	98.8	0.99	98.8	98.8	631
	AdaMax	0.04	99.9	1.00	99.9	99.9	0.06	98.9	0.99	98.9	98.9	714
	AdaGrad	0.04	99.9	1.00	99.9	99.9	0.07	98.7	0.99	98.7	98.7	681
Glorot (Xavier) Normal	Adam	0.07	99.6	1.00	99.6	99.6	0.11	98.6	0.99	98.6	98.5	765
	AdaDelta	0.05	99.8	1.00	99.8	99.8	0.10	98.6	0.99	98.7	98.5	819
	SGD	0.04	99.9	1.00	99.9	99.9	0.07	99.0	0.99	99.1	99.0	651
	AdaMax	0.04	99.9	1.00	99.9	99.9	0.08	99.0	0.99	99.0	99.0	729
	AdaGrad	0.04	99.9	1.00	99.9	99.9	0.08	99.0	0.99	99.0	99.0	714
Glorot (Xavier) Uniform	Adam	0.07	99.5	1.00	99.5	99.5	0.10	98.4	0.98	98.4	98.4	792
	AdaDelta	0.04	99.9	1.00	99.9	99.9	0.08	99.1	0.99	99.1	99.1	853
	SGD	0.04	99.9	1.00	99.9	99.9	0.08	98.7	0.99	98.7	98.7	675
	AdaMax	0.03	99.9	1.00	99.9	99.9	0.09	98.7	0.99	98.7	98.7	756
	AdaGrad	0.03	99.9	1.00	99.9	99.9	0.06	99.1	0.99	99.1	99.1	727
LeCun Normal	Adam	0.07	99.7	1.00	99.7	99.7	0.10	98.9	0.99	98.9	98.9	818
	AdaDelta	0.05	99.8	1.00	99.8	99.8	0.09	99.0	0.99	99.0	99.0	870
	SGD	0.04	99.9	1.00	99.9	99.8	0.08	98.9	0.99	98.9	98.8	702
	AdaMax	0.04	99.9	1.00	99.9	99.8	0.08	99.0	0.99	99.1	98.9	788
	AdaGrad	0.03	99.9	1.00	99.9	99.9	0.07	99.1	0.99	99.1	99.1	753
LeCun Uniform	Adam	0.06	99.7	1.00	99.7	99.7	0.09	98.8	0.99	98.8	98.8	835
	AdaDelta	0.05	99.7	1.00	99.7	99.7	0.10	98.7	0.99	98.7	98.7	892
	SGD	0.04	99.9	1.00	99.9	99.9	0.08	98.5	0.99	98.6	98.5	720
	AdaMax	**0.03**	**99.9**	**1.00**	**99.9**	**99.9**	**0.06**	**99.2**	**0.99**	**99.2**	**99.2**	**803**
	AdaGrad	0.03	99.9	1.00	99.9	99.9	0.07	99.1	0.99	99.1	99.1	772

Table 22 Comparing current study and other authors reported accuracies using CMATER dataset

Authors	Testing accuracy
Current Study	100% (with the HMB1 architecture)
	99.2% and 99.4% (with the HMB2 architecture)
[74]	98.59%
[75]	97.40%

control architecture to work with the study larger and more complex datasets.

5.3.3 Phase 2 (test the described architectures' generalization versus the control dataset)

Tables 28 and 29 show the corresponding results of the different experiments mentioned in Table 6, respectively. They show the loss, accuracy, F1, precision, and recall values for training and testing phases. The training time is reported in seconds. The best results are in bold.

Table 28 shows that the AdaDelta optimizer and LeCun Uniform weight initializer report the highest testing accuracy which is 97.3% using HMB1 in phase two from the cross-validation testing. The average training time is 740 s. The average training and testing accuracies are 99.8%, and 96.1%, respectively. The average training and testing losses are 0.01, and 0.18, respectively. Figure 46 shows the

Table 23 Training and testing reported results of Experiment 13

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.18	94.4	0.94	95.3	93.4	0.42	89.1	0.89	90.2	88.0	303
	AdaDelta	0.02	99.6	1.00	99.6	99.5	0.45	91.5	0.92	91.7	91.3	324
	SGD	0.65	78.8	0.78	85.7	71.2	0.75	74.4	0.73	80.2	66.8	249
	AdaMax	0.11	96.7	0.97	97.3	96.2	0.45	88.2	0.88	89.9	86.6	289
	AdaGrad	0.19	94.2	0.94	95.3	93.2	0.38	88.6	0.88	89.5	85.7	277
He Uniform	Adam	0.17	95.0	0.95	95.9	94.1	0.44	89.5	0.90	91.3	88.2	310
	AdaDelta	**0.02**	**99.5**	**1.00**	**99.5**	**99.5**	**0.42**	**92.7**	**0.93**	**92.9**	**92.7**	**343**
	SGD	0.76	79.0	0.78	86.1	70.7	0.89	75.1	0.75	82.8	68.4	270
	AdaMax	0.10	96.9	0.97	97.3	96.6	0.43	91.8	0.92	92.7	91.3	306
	AdaGrad	0.19	94.1	0.94	95.2	93.2	0.40	88.9	0.88	89.8	86.9	293
Glorot (Xavier) Normal	Adam	0.19	94.0	0.94	95.1	92.9	0.43	87.1	0.87	89.2	85.7	332
	AdaDelta	0.03	99.4	0.99	99.4	99.3	0.45	90.4	0.91	91.4	90.0	360
	SGD	0.58	81.1	0.80	86.7	74.8	0.70	75.7	0.76	81.0	71.5	292
	AdaMax	0.14	95.7	0.96	96.3	95.2	0.34	90.2	0.91	92.4	88.9	329
	AdaGrad	0.22	93.1	0.93	94.5	91.7	0.48	84.6	0.85	87.5	82.4	314
Glorot (Xavier) Uniform	Adam	0.30	89.8	0.90	92.0	87.8	0.50	83.1	0.83	85.9	80.0	354
	AdaDelta	0.02	99.6	1.00	99.6	99.6	0.40	92.2	0.92	92.4	92.0	381
	SGD	0.58	82.6	0.82	88.9	75.8	0.68	79.1	0.77	83.6	71.3	308
	AdaMax	0.12	96.1	0.96	96.6	95.4	0.36	91.3	0.92	93.8	90.4	351
	AdaGrad	0.17	95.0	0.95	96.1	94.1	0.38	89.5	0.90	91.0	88.4	331
LeCun Normal	Adam	0.25	92.0	0.92	93.5	90.6	0.45	85.1	0.85	86.8	82.9	369
	AdaDelta	0.02	99.6	1.00	99.6	99.6	0.44	92.4	0.92	92.8	92.0	396
	SGD	0.61	80.2	0.79	86.2	73.3	0.72	76.4	0.77	83.8	71.7	323
	AdaMax	0.10	96.9	0.97	97.2	96.6	0.33	92.0	0.92	92.6	90.6	367
	AdaGrad	0.24	92.6	0.93	94.4	91.1	0.43	86.0	0.87	88.5	84.9	349
LeCun Uniform	Adam	0.30	90.1	0.90	92.0	88.4	0.46	84.2	0.85	87.3	82.2	387
	AdaDelta	0.02	99.8	1.00	99.8	99.8	0.47	92.4	0.93	93.2	92.2	415
	SGD	0.47	85.3	0.85	89.2	80.9	0.61	81.5	0.81	86.1	76.6	348
	AdaMax	0.13	96.2	0.96	96.6	95.6	0.38	89.5	0.90	90.6	89.1	390
	AdaGrad	0.32	90.1	0.90	92.5	87.4	0.47	86.6	0.86	89.1	83.5	369

experiment training and testing accuracies curves with their trendlines. Figure 47 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

Table 29 shows that the AdaMax optimizer and LeCun Uniform weight initializer report the highest testing accuracy which is 96.8% using HMB2 in phase two from the cross-validation testing. The average training time is 402 s. The average training and testing accuracies are 99.7%, and 96.1%, respectively. The average training and testing losses are 0.08, and 0.24, respectively. Figure 48 shows the experiment training and testing accuracies curves with their trendlines. Figure 49 shows the training time curve with its trendline. The x-axis refers to the corresponding experiment rows, and the y-axis refers to the accuracy as a percentage (in the accuracies figure) and refers to the time in seconds (in the time figure).

5.3.4 Comments on the second phase

From the obtained results, the study described architectures generalize for the control dataset and have higher accuracies, than the reported accuracy in the selected control study for the different comparing tests.

Table 30 summarizes the obtained results in the two phases.

Table 24 Training and testing reported results of Experiment 14

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.15	99.0	0.99	99.0	99.0	0.61	92.0	0.92	92.0	92.0	161
	AdaDelta	0.08	99.6	1.00	99.6	99.6	0.47	92.9	0.93	93.1	92.9	152
	SGD	0.06	99.5	1.00	99.5	99.5	0.41	91.8	0.92	92.3	91.3	171
	AdaMax	0.08	99.5	1.00	99.5	99.5	0.48	92.7	0.93	92.8	92.4	182
	AdaGrad	0.06	99.7	1.00	99.7	99.6	0.41	92.0	0.92	92.8	92.0	150
He Uniform	Adam	0.14	99.2	0.99	99.2	99.1	0.60	92.2	0.92	92.4	91.8	164
	AdaDelta	0.08	99.7	1.00	99.7	99.7	0.58	92.0	0.92	92.4	92.0	159
	SGD	0.05	99.7	1.00	99.7	99.6	0.40	91.3	0.91	91.3	90.9	177
	AdaMax	0.07	99.7	1.00	99.7	99.6	0.47	91.5	0.92	91.7	91.3	188
	AdaGrad	**0.06**	**99.7**	**1.00**	**99.7**	**99.7**	**0.38**	**93.3**	**0.93**	**93.5**	**93.1**	**157**
Glorot (Xavier) Normal	Adam	0.15	99.3	0.99	99.3	99.2	0.59	93.3	0.94	93.7	93.3	175
	AdaDelta	0.08	99.6	1.00	99.7	99.6	0.44	92.0	0.92	92.2	92.0	166
	SGD	0.06	99.6	1.00	99.7	99.6	0.37	92.4	0.92	92.5	92.0	188
	AdaMax	0.07	99.7	1.00	99.7	99.7	0.46	92.0	0.92	92.4	92.0	198
	AdaGrad	0.06	99.7	1.00	99.7	99.7	0.41	91.8	0.92	92.0	91.8	167
Glorot (Xavier) Uniform	Adam	0.16	98.7	0.99	98.7	98.6	0.62	90.6	0.91	90.8	90.6	183
	AdaDelta	0.07	99.6	1.00	99.6	99.6	0.59	91.5	0.92	91.9	91.5	175
	SGD	0.06	99.5	1.00	99.5	99.5	0.41	91.3	0.91	91.9	90.6	196
	AdaMax	0.17	96.2	0.96	96.8	95.7	0.38	91.3	0.91	92.0	90.6	208
	AdaGrad	0.07	99.6	1.00	99.6	99.5	0.38	91.1	0.91	91.1	91.1	177
LeCun Normal	Adam	0.13	99.0	0.99	99.0	98.9	0.58	91.3	0.92	92.3	91.1	193
	AdaDelta	0.07	99.6	1.00	99.6	99.6	0.51	91.3	0.91	91.3	91.3	187
	SGD	0.06	99.4	0.99	99.4	99.3	0.36	91.3	0.92	91.9	91.1	207
	AdaMax	0.07	99.6	1.00	99.6	99.5	0.46	91.3	0.92	92.1	91.1	219
	AdaGrad	0.06	99.6	1.00	99.7	99.6	0.38	92.9	0.93	93.1	92.9	186
LeCun Uniform	Adam	0.15	99.5	1.00	99.5	99.5	0.62	91.8	0.92	92.2	91.8	205
	AdaDelta	0.07	99.7	1.00	99.7	99.7	0.49	92.7	0.93	93.6	92.2	195
	SGD	0.06	99.6	1.00	99.6	99.5	0.38	91.5	0.92	92.1	91.1	161
	AdaMax	0.07	99.6	1.00	99.7	99.6	0.41	92.0	0.92	92.3	92.0	152
	AdaGrad	0.06	99.7	1.00	99.7	99.7	0.39	91.3	0.91	91.7	90.6	171

6 Conclusions

The current research presented a valid system for Arabic handwritten character recognition with the ability to select between two CNN architectures, with a large, and complex dataset for the AHCR problem. The reason for presenting two CNN architectures was to study the effects of changing the architecture complexity. As discussed in Sect. 4.1, HMB1 was more complex (more trainable parameters) than HMB2. However, HMB2 used regularization to study the effect of reducing the complexity with the application of the overfitting reduction technique (regularization). Table 31 summarizes the different 16 experiments with the highest testing accuracies with the corresponding best weights initializers and optimizers.

From the obtained results, we can conclude that data augmentation helped to improve the testing accuracy and decrease the overfitting. Data augmentation increased the number of inputs data and hence the architectures learned and trained on more data. HMB1 reported higher

Table 25 Training and testing reported results of Experiment 15

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.01	99.9	1.00	99.9	99.9	0.05	98.7	0.99	98.7	98.7	3152
	AdaDelta	0.00	99.9	1.00	99.9	99.9	0.07	98.7	0.99	98.7	98.7	2956
	SGD	0.01	99.9	1.00	99.9	99.9	0.09	98.2	0.98	98.2	98.1	3270
	AdaMax	0.00	99.9	1.00	99.9	99.9	0.08	98.6	0.99	98.6	98.6	3315
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.11	98.0	0.98	98.1	98.0	2582
He Uniform	Adam	0.01	99.9	1.00	99.9	99.9	0.06	98.8	0.99	98.9	98.8	2924
	AdaDelta	0.00	99.9	1.00	99.9	99.9	0.07	98.9	0.99	99.0	98.9	2656
	SGD	0.00	99.9	1.00	99.9	99.9	0.09	98.2	0.98	98.3	98.2	2938
	AdaMax	**0.00**	**99.9**	**1.00**	**99.9**	**99.9**	**0.06**	**99.0**	**0.99**	**99.0**	**99.0**	**3240**
	AdaGrad	0.00	99.9	1.00	99.9	99.9	0.10	98.4	0.98	98.4	98.3	2565
Glorot (Xavier) Normal	Adam	0.01	99.9	1.00	99.9	99.9	0.08	98.6	0.99	98.6	98.6	2946
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.08	98.9	0.99	98.9	98.9	3178
	SGD	0.01	99.9	1.00	99.9	99.9	0.07	98.4	0.98	98.4	98.4	3630
	AdaMax	0.00	99.9	1.00	99.9	99.9	0.07	98.8	0.99	98.9	98.8	3881
	AdaGrad	0.00	99.9	1.00	99.9	99.9	0.09	98.4	0.98	98.4	98.4	3027
Glorot (Xavier) Uniform	Adam	0.01	99.9	1.00	99.9	99.9	0.07	98.6	0.99	98.6	98.6	3424
	AdaDelta	0.01	99.9	1.00	99.9	99.9	0.07	98.8	0.99	98.8	98.7	3293
	SGD	0.01	99.9	1.00	99.9	99.9	0.07	98.3	0.98	98.4	98.3	3690
	AdaMax	0.00	99.9	1.00	99.9	99.9	0.07	98.8	0.99	98.8	98.8	3122
	AdaGrad	0.00	99.9	1.00	99.9	99.9	0.09	98.4	0.98	98.4	98.4	2395
LeCun Normal	Adam	0.01	99.9	1.00	99.9	99.9	0.06	98.7	0.99	98.7	98.6	2829
	AdaDelta	0.00	99.9	1.00	99.9	99.9	0.07	98.8	0.99	98.9	98.8	2725
	SGD	0.01	99.9	1.00	99.9	99.9	0.09	98.2	0.98	98.2	98.2	3074
	AdaMax	0.00	99.9	1.00	99.9	99.9	0.07	98.8	0.99	98.8	98.7	3842
	AdaGrad	0.00	99.9	1.00	99.9	99.9	0.08	98.4	0.98	98.5	98.4	2942
LeCun Uniform	Adam	0.01	99.8	1.00	99.8	99.8	0.06	98.7	0.99	98.8	98.7	3334
	AdaDelta	0.00	99.9	1.00	99.9	99.9	0.07	98.9	0.99	98.9	98.9	3160
	SGD	0.01	99.9	1.00	99.9	99.9	0.08	98.4	0.98	98.4	98.4	3152
	AdaMax	0.00	99.9	1.00	99.9	99.9	0.06	98.7	0.99	98.7	98.6	2956
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.08	98.4	0.98	98.4	98.4	3270

accuracies (but not very high) than HMB2 as the first one's architecture is more complex and has more parameters than the later one but requires more time to train. Regularization helped to decrease the overfitting in HMB2 although the architecture is less complex. By increasing the dataset width and height, lead to more parameters and less accuracy and this concludes that, the more the parameters, the more the required data. He Uniform reported the best accuracies in 6 experiments and hence can be considered as a suitable weight initializer to use. AdaDelta optimizer reported the best accuracies in 5 experiments and hence can be considered as a suitable optimizer to use. The average time in each experiment (from 1 to 16) is calculated and plotted in Fig. 50. Experiment 7 had the longest time between the different experiments because it worked on 865,840 elements using the HMBD dataset and HMB1 architecture.

The two described architectures achieved more than 99.2% and 100% using CMATER dataset with and without data augmentation, respectively. HMB1 and HMB2

Table 26 Training and testing reported results of Experiment 16

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.13	98.6	0.99	98.7	98.5	0.17	97.3	0.97	97.6	97.2	1972
	AdaDelta	0.11	99.2	0.99	99.2	99.1	0.15	97.9	0.98	98.1	97.7	1897
	SGD	**0.04**	**99.9**	**1.00**	**99.9**	**99.9**	**0.10**	**98.4**	**0.98**	**98.5**	**98.4**	**2105**
	AdaMax	0.07	99.6	1.00	99.7	99.6	0.13	98.2	0.98	98.3	98.0	2274
	AdaGrad	0.04	99.9	1.00	99.9	99.9	0.10	98.0	0.98	98.1	98.0	1796
He Uniform	Adam	0.13	98.6	0.99	98.7	98.6	0.17	97.3	0.97	97.5	97.1	1585
	AdaDelta	0.13	98.7	0.99	98.7	98.6	0.18	97.3	0.97	97.5	97.2	1510
	SGD	0.04	99.8	1.00	99.8	99.8	0.09	98.3	0.98	98.4	98.2	1717
	AdaMax	0.08	99.6	1.00	99.6	99.6	0.13	98.0	0.98	98.0	98.0	1846
	AdaGrad	0.05	99.8	1.00	99.8	99.8	0.10	98.2	0.98	98.3	98.0	1454
Glorot (Xavier) Normal	Adam	0.13	98.5	0.99	98.5	98.4	0.18	97.3	0.97	97.5	97.2	1626
	AdaDelta	0.12	98.9	0.99	99.0	98.8	0.17	97.2	0.97	97.5	97.0	1554
	SGD	0.04	99.8	1.00	99.8	99.8	0.10	98.2	0.98	98.3	98.1	1696
	AdaMax	0.08	99.5	1.00	99.5	99.5	0.12	98.2	0.98	98.3	98.2	1801
	AdaGrad	0.04	99.9	1.00	99.9	99.9	0.10	98.1	0.98	98.1	98.0	1435
Glorot (Xavier) Uniform	Adam	0.13	98.7	0.99	98.8	98.6	0.18	97.1	0.97	97.3	97.0	1584
	AdaDelta	0.11	99.2	0.99	99.2	99.1	0.16	97.7	0.98	97.9	97.6	1528
	SGD	0.04	99.8	1.00	99.8	99.8	0.10	98.1	0.98	98.1	98.0	1688
	AdaMax	0.08	99.4	0.99	99.4	99.4	0.13	97.7	0.98	97.7	97.7	1811
	AdaGrad	0.04	99.9	1.00	99.9	99.9	0.10	97.9	0.98	98.0	97.9	1474
LeCun Normal	Adam	0.13	98.5	0.99	98.6	98.5	0.18	97.0	0.97	97.1	96.9	1674
	AdaDelta	0.11	99.1	0.99	99.2	99.1	0.16	97.7	0.98	97.9	97.7	1595
	SGD	0.04	99.8	1.00	99.9	99.8	0.11	98.0	0.98	98.0	97.9	1780
	AdaMax	0.07	99.7	1.00	99.7	99.6	0.13	98.1	0.98	98.2	98.1	1921
	AdaGrad	0.04	99.8	1.00	99.8	99.8	0.11	97.5	0.98	97.6	97.5	1737
LeCun Uniform	Adam	0.13	98.5	0.99	98.6	98.4	0.17	97.4	0.98	97.6	97.3	1939
	AdaDelta	0.12	99.0	0.99	99.0	98.9	0.17	97.6	0.98	97.7	97.5	1940
	SGD	0.04	99.7	1.00	99.8	99.7	0.11	98.0	0.98	98.2	97.9	1972
	AdaMax	0.07	99.6	1.00	99.6	99.6	0.13	98.0	0.98	98.1	98.0	1897
	AdaGrad	0.04	99.9	1.00	99.9	99.8	0.10	98.0	0.98	98.1	98.0	2105

Fig. 13 Experiment 1 training and testing accuracies curves with their trendlines

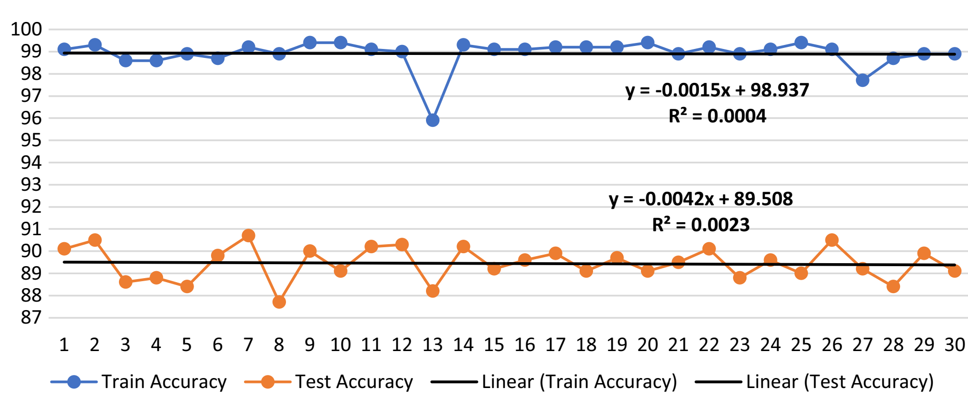


Fig. 14 Experiment 1 Training Time Curve with its Trendline

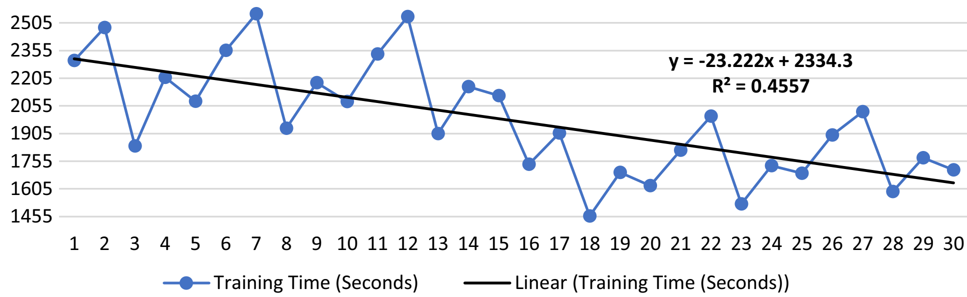


Fig. 15 Experiment 2 training and testing accuracies curves with their trendlines

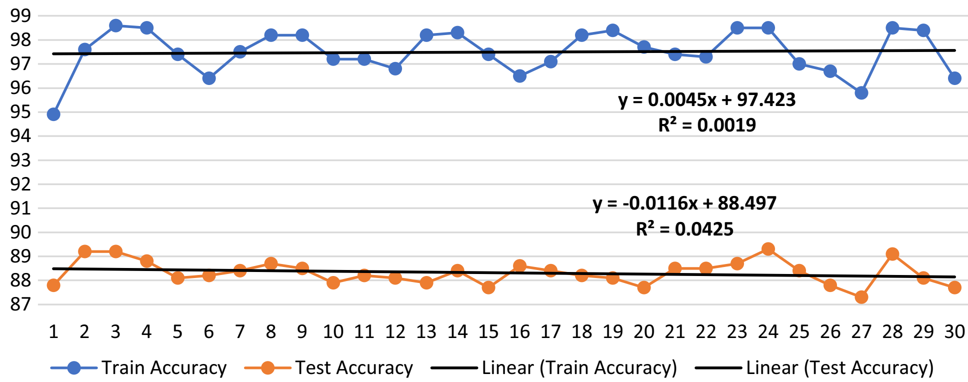


Fig. 16 Experiment 2 training time curve with its trendline

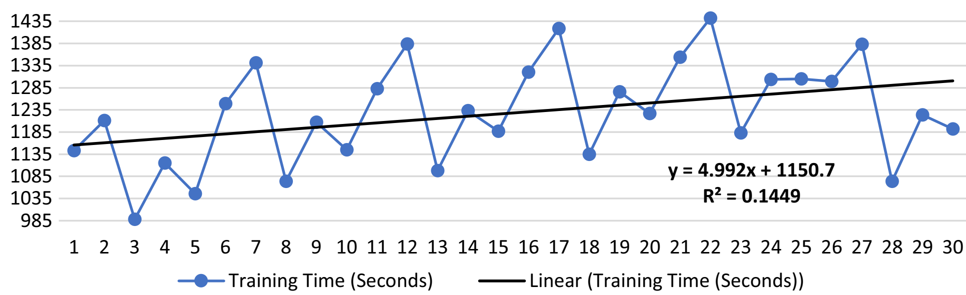


Fig. 17 Experiment 3 training and testing accuracies curves with their trendlines

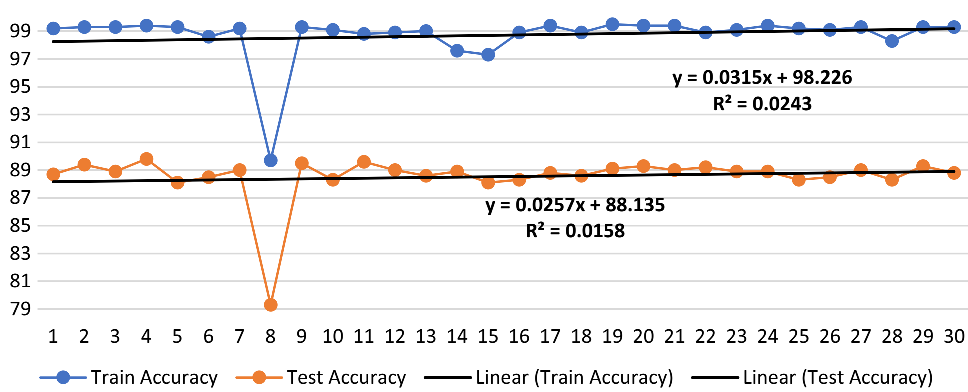


Fig. 18 Experiment 3 training time curve with its trendline

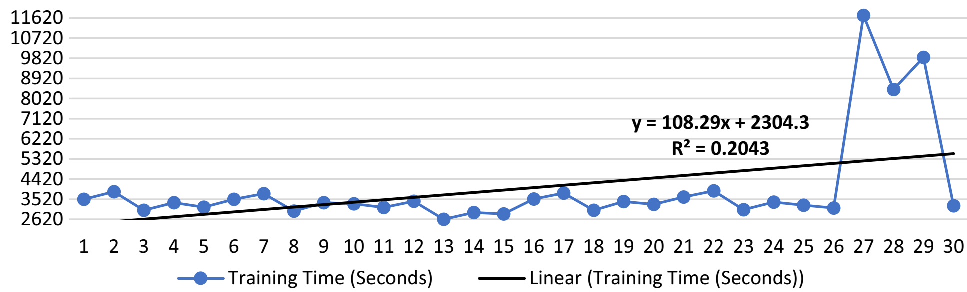


Fig. 19 Experiment 4 training and testing accuracies curves with their trendlines

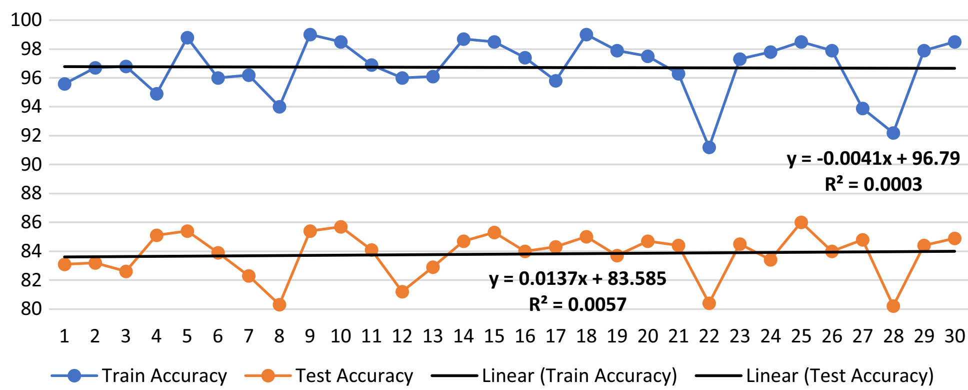


Fig. 20 Experiment 4 training time curve with its trendline

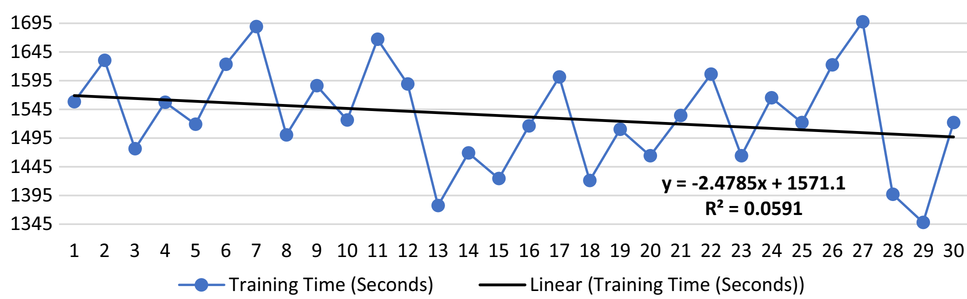
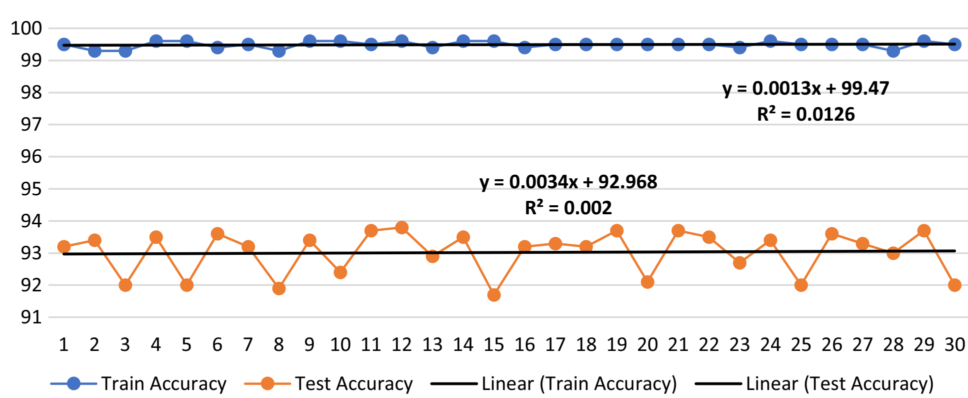


Fig. 21 Experiment 5 training and testing accuracies curves with their trendlines



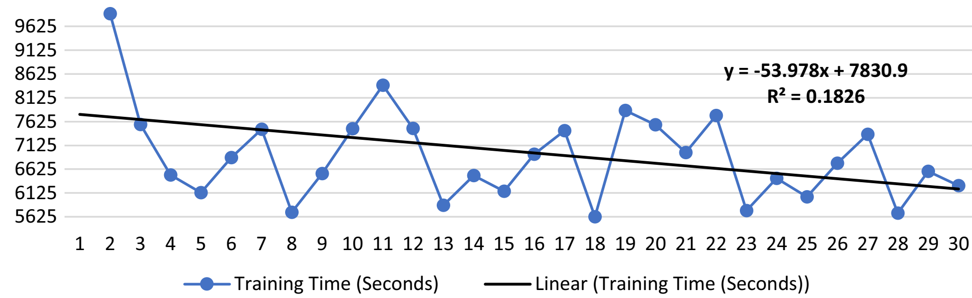


Fig. 22 Experiment 5 training time curve with its trendline

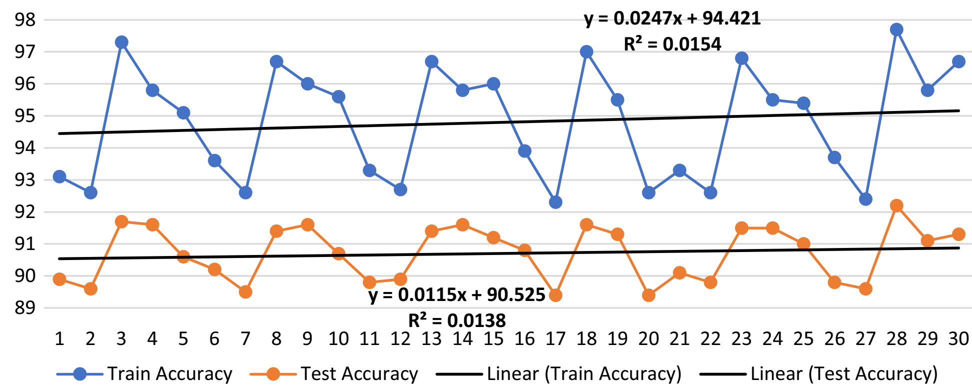


Fig. 23 Experiment 6 training and testing accuracies curves with their trendlines

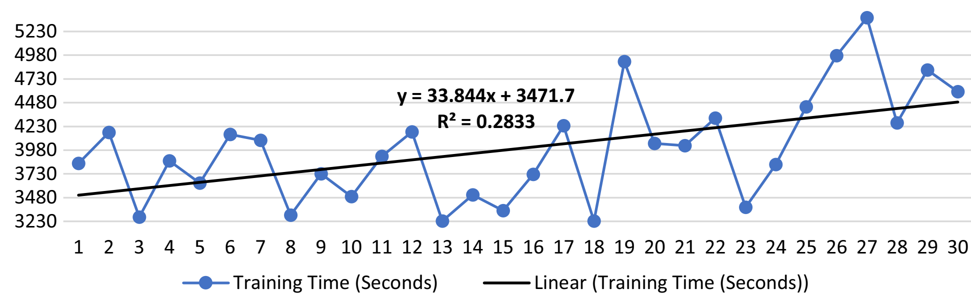


Fig. 24 Experiment 6 training time curve with its trendline

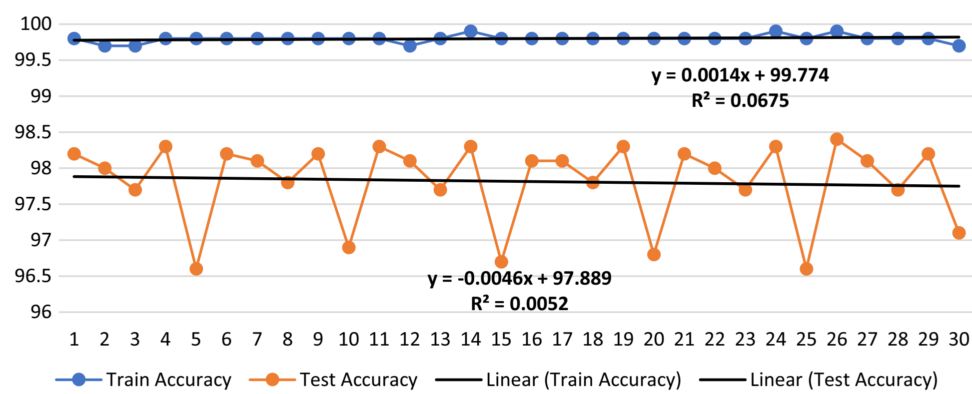


Fig. 25 Experiment 7 training and testing accuracies curves with their trendlines

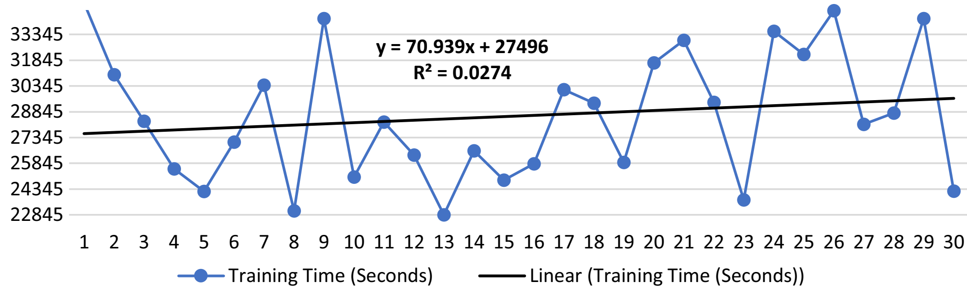


Fig. 26 Experiment 7 training time curve with its trendline

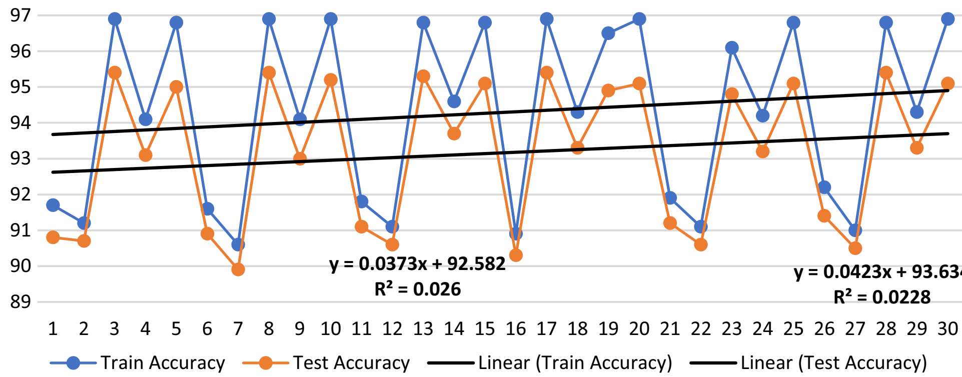


Fig. 27 Experiment 8 training and testing accuracies curves with their trendlines

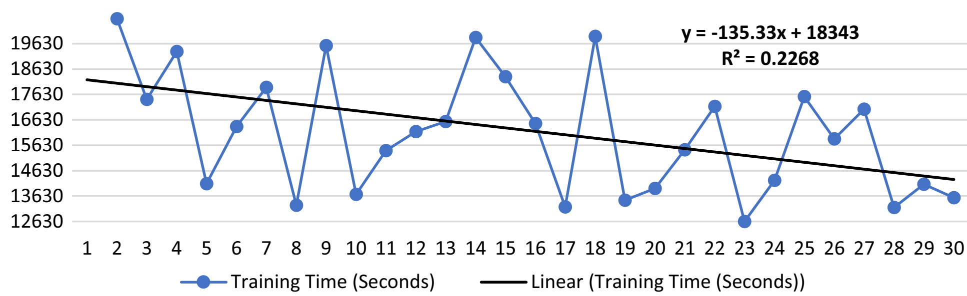


Fig. 28 Experiment 8 training time curve with its trendline

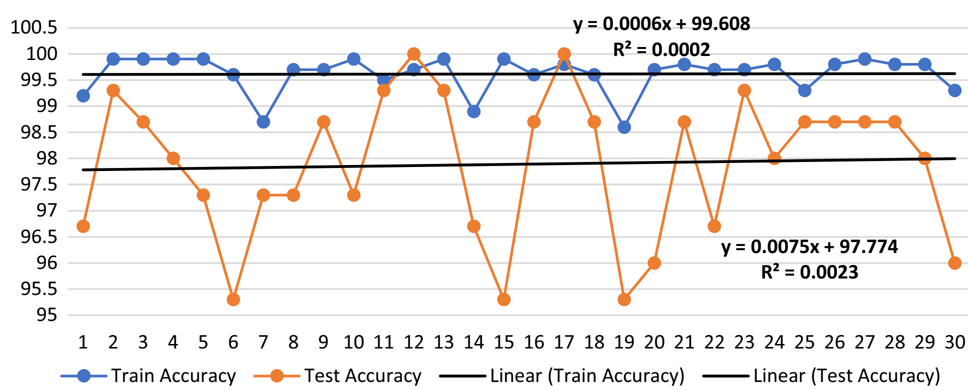


Fig. 29 Experiment 9 training and testing accuracies curves with their trendlines

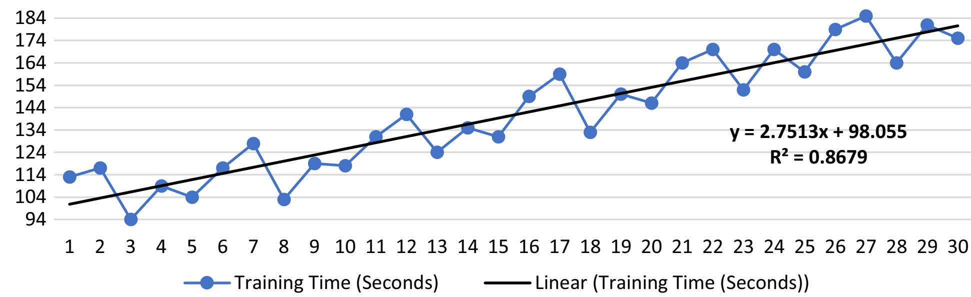


Fig. 30 Experiment 9 training time curve with its trendline

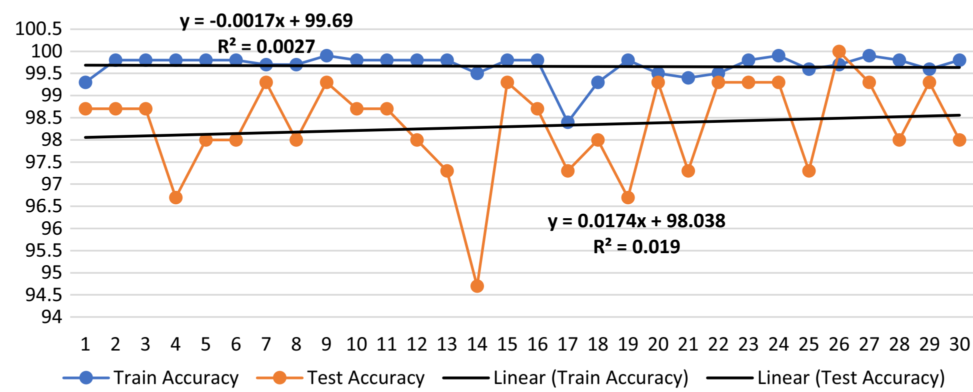


Fig. 31 Experiment 10 training and testing accuracies curves with their trendlines

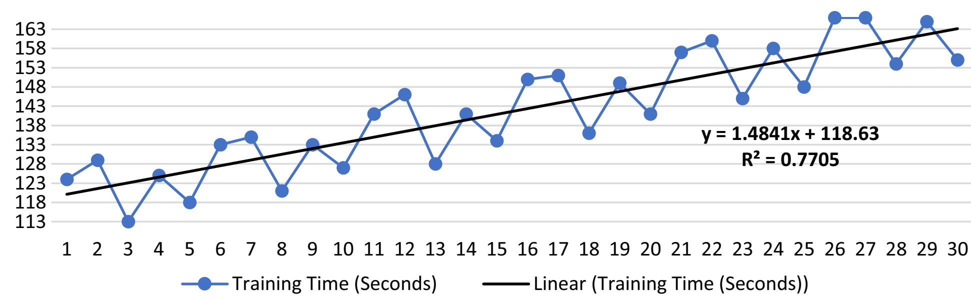


Fig. 32 Experiment 10 training time curve with its trendline

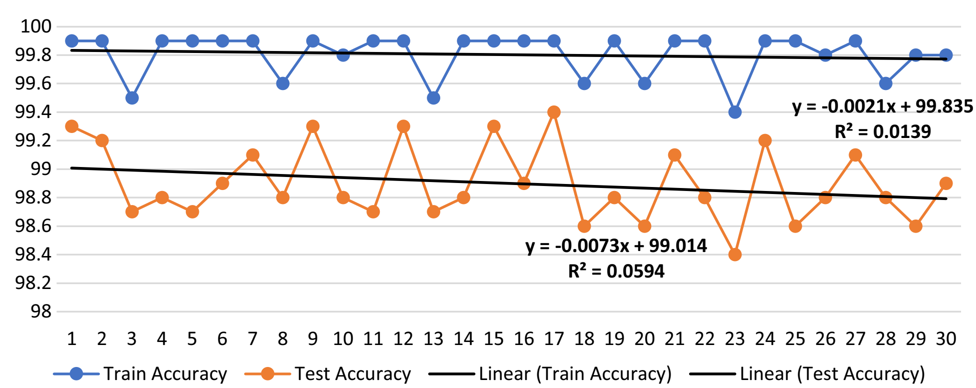


Fig. 33 Experiment 11 training and testing accuracies curves with their trendlines

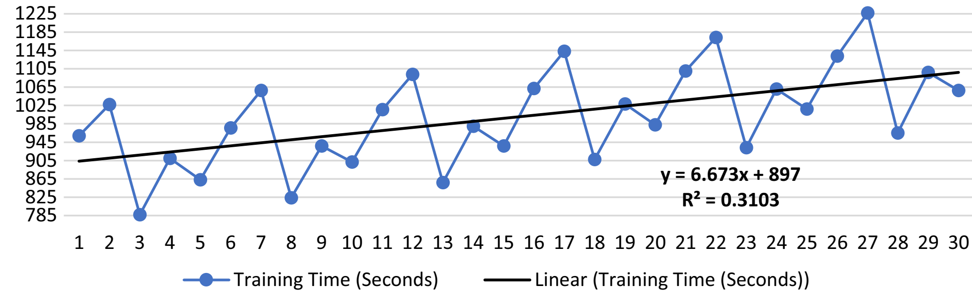


Fig. 34 Experiment 11 training time curve with its trendline

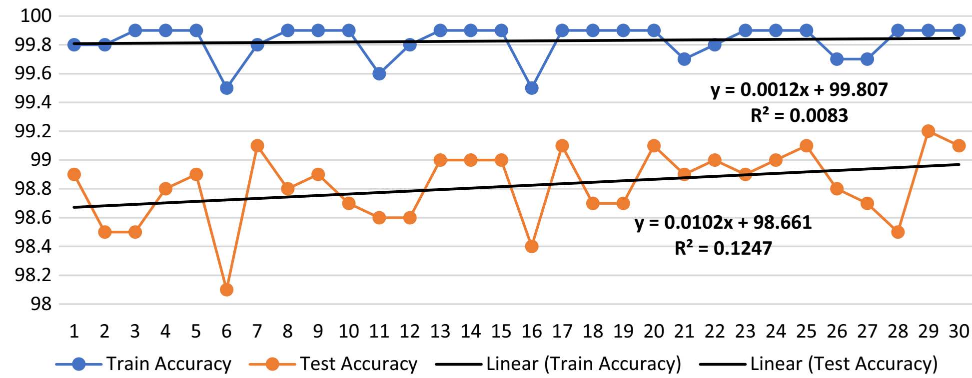


Fig. 35 Experiment 12 training and testing accuracies curves with their trendlines

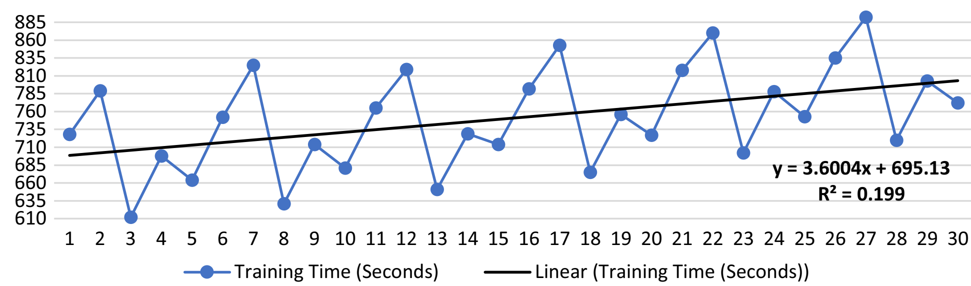


Fig. 36 Experiment 12 training time curve with its trendline

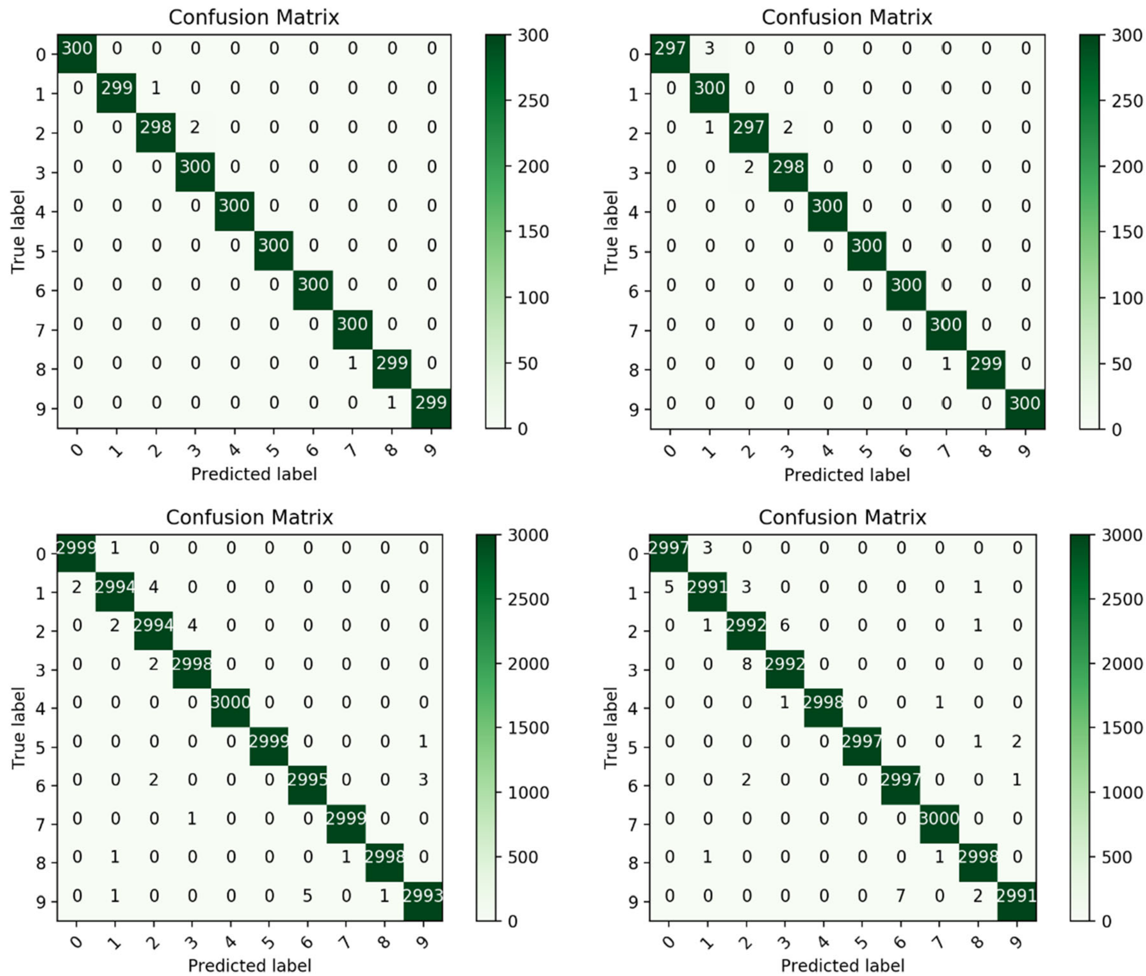
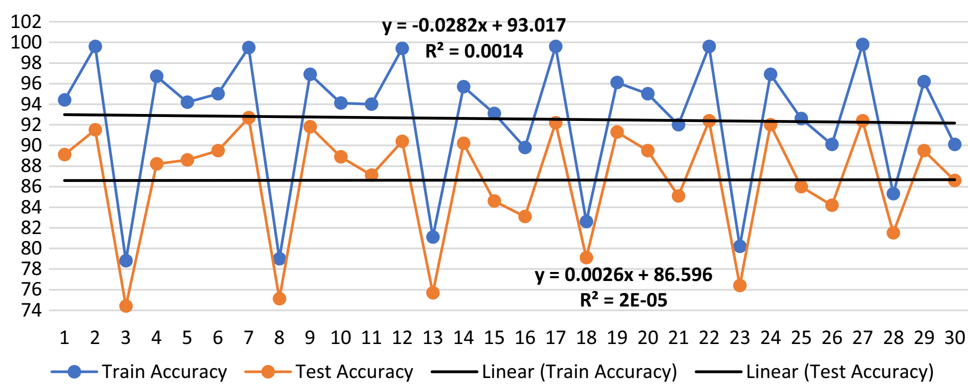


Fig. 37 Confusion matrices (top left, top right, bottom left, and bottom right) for Experiments 9 to 12, respectively

Fig. 38 Experiment 13 training and testing accuracies curves with their trendlines



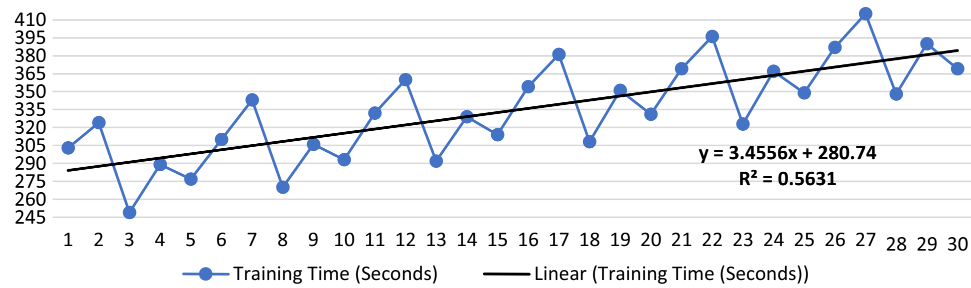


Fig. 39 Experiment 13 training time curve with its trendline

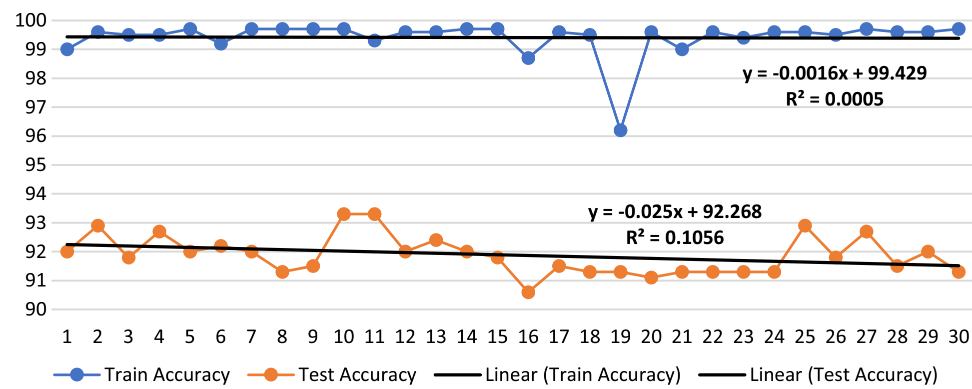


Fig. 40 Experiment 14 training and testing accuracies curves with their trendlines

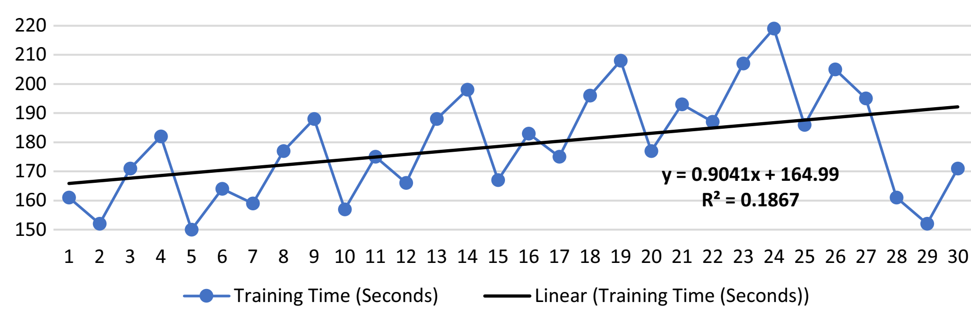


Fig. 41 Experiment 14 training time curve with its trendline

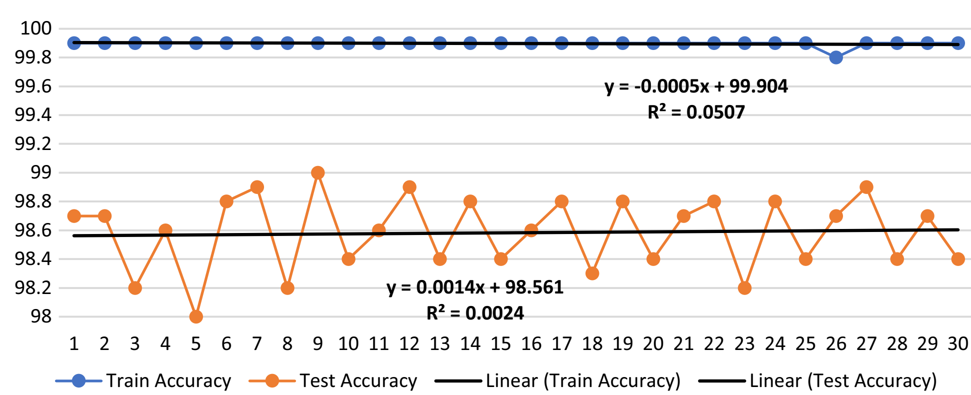


Fig. 42 Experiment 15 training and testing accuracies curves with their trendlines

Fig. 43 Experiment 15 training time curve with its trendline

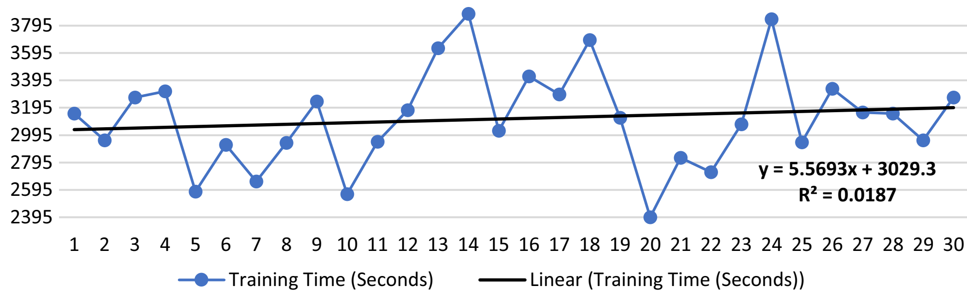


Fig. 44 Experiment 16 training and testing accuracies curves with their trendlines

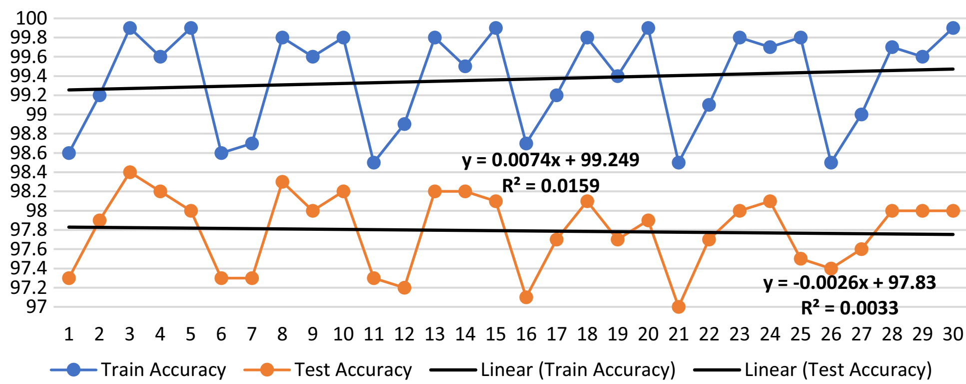


Fig. 45 Experiment 16 training time curve with its trendline

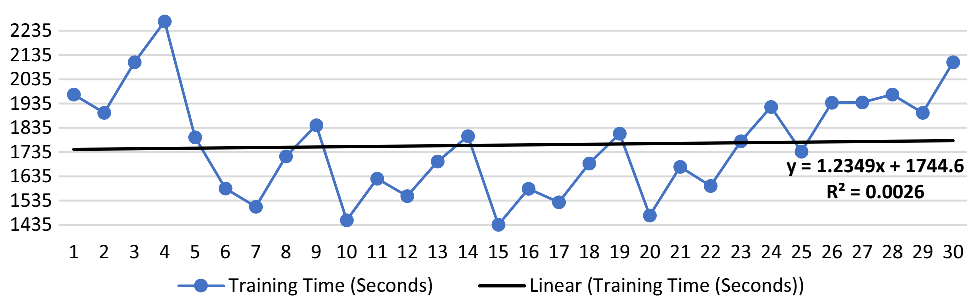


Table 27 Results obtained from testing the described architecture in [27] on the presented dataset

Weight initializer	Training phase					Testing phase				
	Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall
He Normal	2.39	41.5	0.39	63.2	28.2	2.52	39.2	0.37	60.3	26.8
He Uniform	2.14	47.6	0.44	63.5	34.0	2.25	44.9	0.42	60.7	32.2
Glorot (Xavier) Normal	2.12	43.5	0.38	72.7	25.9	2.25	39.9	0.34	68.2	22.9
Glorot (Xavier) Uniform	**1.84**	**51.0**	**0.49**	**72.5**	**37.2**	**2.03**	**47.7**	**0.46**	**67.8**	**35.0**
LeCun Normal	2.93	36.8	0.35	51.4	26.5	3.03	35.5	0.33	49.5	25.0
LeCun Uniform	2.56	42.4	0.42	56.1	33.9	2.76	40.5	0.40	53.3	31.9

Table 28 Phase 2 results using the described HMB1 and their control dataset

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.02	99.4	0.99	99.4	99.4	0.17	96.1	0.96	96.3	96.1	671
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.18	96.5	0.97	96.5	96.4	640
	SGD	0.01	99.8	1.00	99.8	99.8	0.21	95.4	0.95	95.8	94.9	719
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.18	95.7	0.96	95.8	95.6	779
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.21	94.9	0.95	94.9	94.9	630
He Uniform	Adam	0.02	99.6	1.00	99.6	99.6	0.21	96.1	0.96	96.3	96.0	693
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.23	95.6	0.96	95.6	95.5	665
	SGD	0.01	99.8	1.00	99.8	99.8	0.22	95.5	0.96	95.8	95.4	744
	AdaMax	0.01	99.7	1.00	99.7	99.7	0.16	96.8	0.97	96.8	96.8	786
	AdaGrad	0.01	99.8	1.00	99.8	99.8	0.16	96.3	0.97	97.0	96.1	651
Glorot (Xavier) Normal	Adam	0.02	99.7	1.00	99.7	99.7	0.18	96.5	0.97	96.7	96.5	722
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.19	96.1	0.96	96.3	96.0	695
	SGD	0.01	99.9	1.00	99.9	99.9	0.18	96.1	0.96	96.2	96.0	782
	AdaMax	0.02	99.3	0.99	99.3	99.2	0.15	95.8	0.96	96.1	95.6	833
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.21	95.8	0.96	95.8	95.8	691
Glorot (Xavier) Uniform	Adam	0.02	99.6	1.00	99.6	99.6	0.18	95.8	0.96	96.0	95.6	761
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.15	96.9	0.97	97.4	96.8	731
	SGD	0.01	99.9	1.00	99.9	99.9	0.15	96.4	0.97	96.9	96.4	810
	AdaMax	0.01	99.8	1.00	99.8	99.8	0.18	96.5	0.97	96.5	96.4	857
	AdaGrad	0.01	99.9	1.00	99.9	99.8	0.20	95.5	0.96	95.9	95.4	713
LeCun Normal	Adam	0.01	99.6	1.00	99.6	99.6	0.21	95.5	0.96	95.8	95.4	784
	AdaDelta	0.01	99.8	1.00	99.8	99.8	0.21	95.8	0.96	96.1	95.8	747
	SGD	0.01	99.9	1.00	99.9	99.8	0.15	96.4	0.97	96.8	96.3	826
	AdaMax	0.01	99.7	1.00	99.7	99.6	0.17	96.1	0.96	96.2	95.7	881
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.19	95.7	0.96	96.2	95.5	736
LeCun Uniform	Adam	0.01	99.7	1.00	99.7	99.7	0.17	96.2	0.96	96.2	96.2	821
	AdaDelta	**0.01**	**99.9**	**1.00**	**99.9**	**99.9**	**0.16**	**97.3**	**0.97**	**97.4**	**97.0**	**793**
	SGD	0.01	99.9	1.00	99.9	99.8	0.19	96.2	0.96	96.6	96.1	671
	AdaMax	0.01	99.8	1.00	99.8	99.7	0.19	96.2	0.96	96.6	96.2	640
	AdaGrad	0.01	99.9	1.00	99.9	99.9	0.19	96.0	0.96	96.2	96.0	719

achieved also more than 92.7% and 98.4% using the AIA9k dataset with and without data augmentation, respectively. The AIA9k dataset achieved less accuracy that the CMATER dataset because of the nature of the input data. The AIA9k dataset included the characters while CMATER dataset included the digits. This reduction can be increased by applying more data augmentation, more regularization, and more training iterations (epochs).

Table 29 Phase 2 results using the described HMB2 and their control dataset

Weight initializer	Optimizer	Training phase					Testing phase					Time
		Loss	Accuracy	F1	Precision	Recall	Loss	Accuracy	F1	Precision	Recall	
He Normal	Adam	0.16	99.4	0.99	99.4	99.4	0.29	96.4	0.96	96.7	96.2	377
	AdaDelta	0.09	99.7	1.00	99.7	99.7	0.24	96.0	0.96	96.1	95.6	357
	SGD	0.04	99.8	1.00	99.8	99.8	0.21	96.3	0.96	96.4	96.2	408
	AdaMax	0.06	99.9	1.00	99.9	99.9	0.23	96.0	0.96	95.9	95.8	433
	AdaGrad	0.05	99.8	1.00	99.8	99.8	0.21	96.3	0.96	96.5	96.2	348
He Uniform	Adam	0.15	99.4	0.99	99.5	99.4	0.34	94.6	0.95	95.0	94.6	392
	AdaDelta	0.08	99.8	1.00	99.8	99.8	0.26	96.3	0.96	96.7	96.2	373
	SGD	0.05	99.5	1.00	99.5	99.5	0.22	95.4	0.96	95.6	95.4	425
	AdaMax	0.06	99.8	1.00	99.8	99.8	0.24	95.5	0.96	95.7	95.4	451
	AdaGrad	0.05	99.9	1.00	99.9	99.9	0.19	95.2	0.95	95.6	95.2	368
Glorot (Xavier) Normal	Adam	0.14	99.5	1.00	99.5	99.5	0.32	96.3	0.96	96.3	96.1	407
	AdaDelta	0.08	99.8	1.00	99.8	99.8	0.24	96.3	0.96	96.8	96.0	387
	SGD	0.04	99.9	1.00	99.9	99.8	0.21	96.7	0.97	96.8	96.4	434
	AdaMax	0.07	99.8	1.00	99.8	99.8	0.19	96.7	0.97	96.8	96.5	467
	AdaGrad	0.06	99.8	1.00	99.8	99.8	0.20	95.8	0.96	96.1	95.8	383
Glorot (Xavier) Uniform	Adam	0.15	99.5	1.00	99.6	99.5	0.36	95.0	0.95	95.4	94.8	422
	AdaDelta	0.08	99.8	1.00	99.8	99.8	0.23	96.7	0.97	96.7	96.5	402
	SGD	0.05	99.8	1.00	99.8	99.8	0.20	96.3	0.96	96.5	96.1	453
	AdaMax	0.06	99.9	1.00	99.8	99.8	0.26	96.2	0.96	96.3	96.1	473
	AdaGrad	0.05	99.8	1.00	99.8	99.8	0.22	96.0	0.96	96.1	95.8	392
LeCun Normal	Adam	0.16	99.5	1.00	99.5	99.5	0.29	96.2	0.96	96.4	95.8	438
	AdaDelta	0.09	99.6	1.00	99.6	99.6	0.27	96.1	0.96	96.3	96.0	418
	SGD	0.04	99.8	1.00	99.8	99.8	0.18	96.2	0.96	96.5	96.1	400
	AdaMax	0.06	99.9	1.00	99.9	99.9	0.22	96.7	0.97	97.0	96.5	422
	AdaGrad	0.05	99.8	1.00	99.8	99.8	0.19	96.2	0.96	96.6	95.8	339
LeCun Uniform	Adam	0.16	99.4	0.99	99.4	99.3	0.33	95.7	0.96	95.9	95.6	380
	AdaDelta	0.08	99.8	1.00	99.8	99.8	0.25	96.2	0.96	96.5	96.2	361
	SGD	0.06	99.5	1.00	99.5	99.4	0.20	95.7	0.96	96.0	95.5	377
	AdaMax	**0.06**	**99.9**	**1.00**	**99.9**	**99.9**	**0.22**	**96.8**	**0.97**	**96.9**	**96.7**	**357**
	AdaGrad	0.05	99.9	1.00	99.9	99.9	0.19	96.7	0.97	97.0	96.7	408

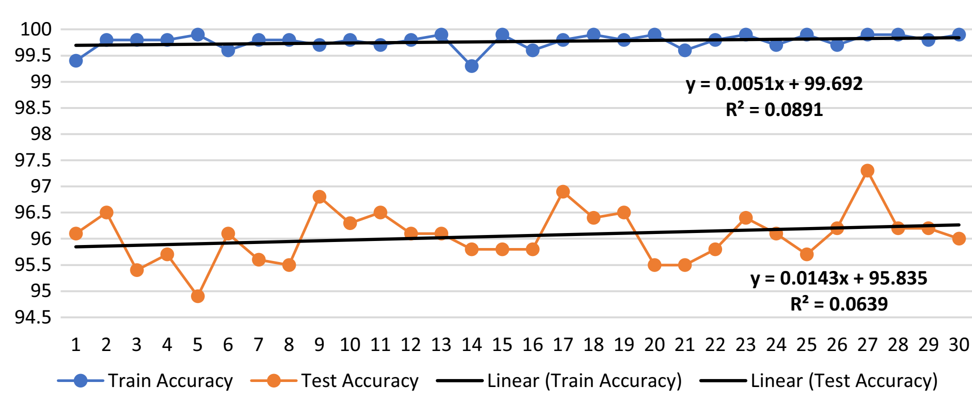


Fig. 46 Phase 2 training and testing accuracies curves with their trendlines using HMB1

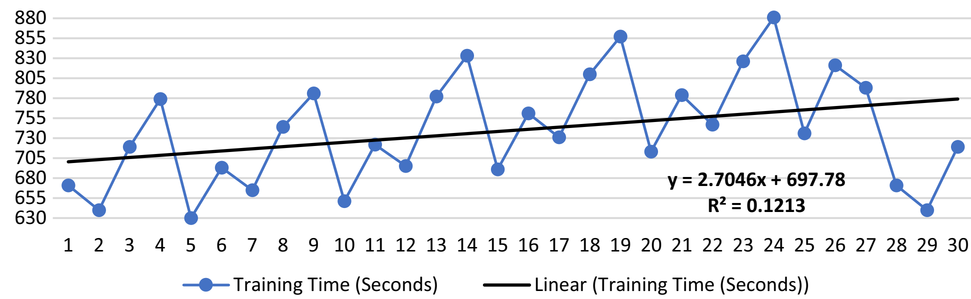


Fig. 47 Phase 2 training time curve with its trendline using HMB1

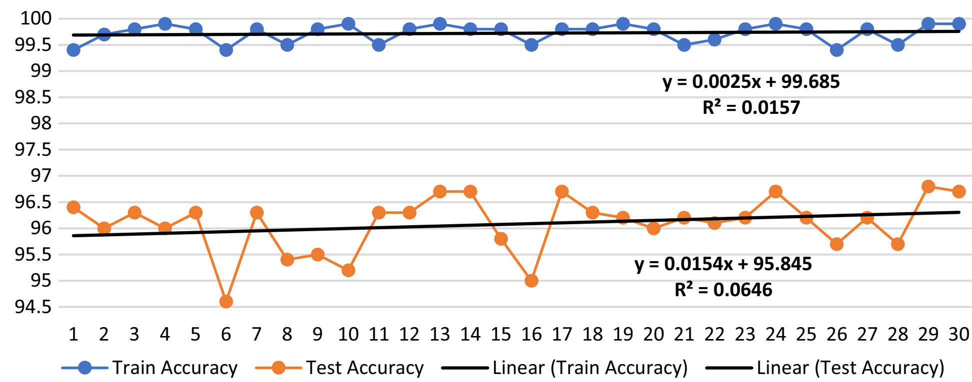


Fig. 48 Phase 2 training and testing accuracies curves with their trendlines using HMB2

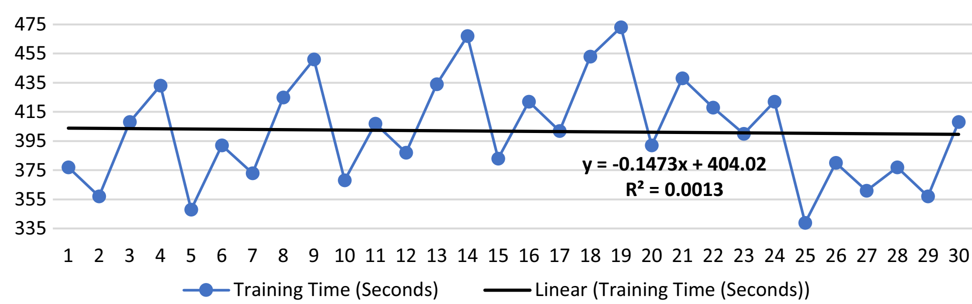


Fig. 49 Phase 2 training time curve with its trendline using HMB1

Table 30 Summarization of the obtained results in the two phases

Phase	Architecture	Dataset	Size	Weight initializer	Optimizer	Testing accuracy (%)
1	[24]	HMBD	54,114	Glorot (Xavier) Uniform	SGD	47.7
	HMB1			He Uniform	AdaDelta	90.7
	HMB2			LeCun Normal	AdaMax	89.3
2	[24] (Reported)	AHCD1 [24]	16,800	–	SGD	94.9
	HMB1			LeCun Uniform	AdaDelta	97.3
	HMB2			LeCun Uniform	AdaMax	96.8

Table 31 Summary of the different performed 16 Experiments

#	Arch.	Dataset	Size	Image Size	Weight initializer	Optimizer	Train accuracy	Test accuracy	Difference (%)
1	HMB1	HMBD	54,114	32×32	**He Uniform**	**AdaDelta**	99.2	90.7	8.5
2	HMB2	HMBD	54,114	32×32	LeCun Normal	AdaMax	98.5	89.3	9.2
3	HMB1	HMBD	54,114	64×64	**He Uniform**	AdaMax	99.4	89.8	9.6
4	HMB2	HMBD	54,114	64×64	LeCun Normal	AdaGrad	98.5	86.0	12.4
5	HMB1	HMBD	216,460	32×32	Glorot (Xavier) Normal	**AdaDelta**	99.6	93.8	5.8
6	HMB2	HMBD	216,460	32×32	LeCun Uniform	SGD	97.7	92.2	5.5
7	HMB1	HMBD	865,840	32×32	LeCun Uniform	Adam	99.9	98.4	1.5
8	HMB2	HMBD	865,840	32×32	**He Uniform**	SGD	96.9	95.4	1.4
9	HMB1	CMATER	3000	32×32	Glorot (Xavier) Uniform	**AdaDelta**	99.8	100.0	−0.2
10	HMB2	CMATER	3000	32×32	LeCun Uniform	Adam	99.7	100.0	−0.3
11	HMB1	CMATER	30,000	32×32	Glorot (Xavier) Uniform	**AdaDelta**	99.9	99.4	0.5
12	HMB2	CMATER	30,000	32×32	LeCun Uniform	AdaMax	99.9	99.2	0.7
13	HMB1	AIA9k	8974	32×32	**He Uniform**	**AdaDelta**	99.5	92.7	6.8
14	HMB2	AIA9k	8974	32×32	**He Uniform**	AdaGrad	99.7	93.3	6.4
15	HMB1	AIA9k	89,740	32×32	**He Uniform**	AdaMax	99.9	99.0	1.0
16	HMB2	AIA9k	89,740	32×32	He Normal	SGD	99.9	98.4	1.4

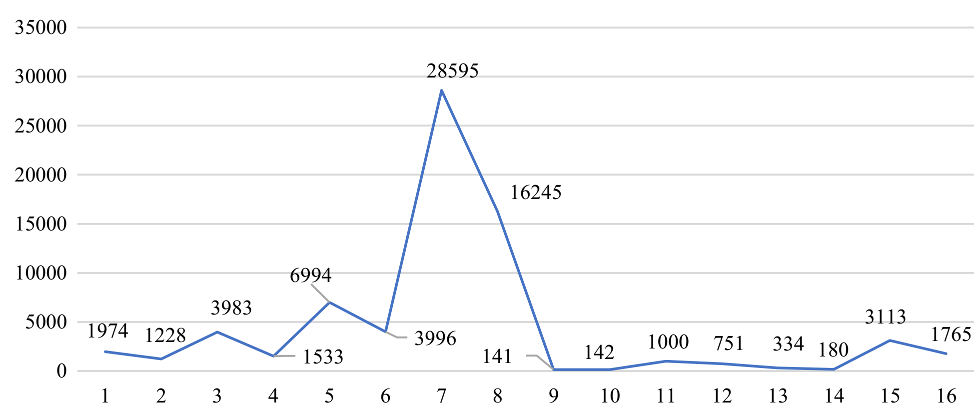


Fig. 50 Average training time in Experiments 1 to 16

From the cross-validation testing, the presented dataset was large enough, so that a previous small architecture could not generalize for it. The described architecture could generalize with a previously published dataset with higher accuracy metrics.

Acknowledgements We would like to express gratitude and appreciation to Prof. Dr. Magdy H. Balaha, who provided guidance, and assistance in this research work and to the Mansoura university volunteers who decided to cooperate in the dataset construction.

References

1. Ridout S (2019) Complete list of Arabic speaking countries—2020 update. http://istizada.com/complete-list-of-arabic-speaking-countries-2014/. Accessed 18-12-2019
2. Versteegh K (2014) Arabic language. Edinburgh University Press, Edinburgh
3. Suleiman Y (2003) The Arabic language and national identity. Edinburgh University Press, Edinburgh
4. Shaalan K, Al-Sheikh S, Oroumchian F (2012) Query expansion based-on similarity of terms for improving Arabic information retrieval. In: International conference on intelligent information processing, pp 167–176
5. El-Desouky AI, Salem MM, El-Gwad AOA, Arafat H (1991) A handwritten Arabic character recognition technique for machine reader. In: Third international conference on software engineering for real time systems, pp 212–216
6. Shirko O, Omar N, Arshad H, Albared M (2010) Machine translation of noun phrases from Arabic to English using transfer-based approach. J Comput Sci 6:350
7. Klatt DH (1987) Review of text-to-speech conversion for English. J Acoust Soc Am 82:737–793
8. Bijl D, Hyde-Thomson H (2001) Speech to text conversion. Google Patents
9. Ashiquzzaman A, Tushar AK, Rahman A, Mohsin F (2019) An efficient recognition method for handwritten arabic numerals using CNN with data augmentation and dropout. In: Data management, analytics and innovation. Springer, 2019, pp 299–309

10. Deng D, Liu H, Li X, Cai D (2018) Pixellink: detecting scene text via instance segmentation. In: Thirty-second AAAI conference on artificial intelligence
11. Korns MF, May T (2019) Strong typing, swarm enhancement, and deep learning feature selection in the pursuit of symbolic regression-classification. In: Genetic programming theory and practice XVI. Springer, pp 59–84
12. Howard J, Ruder S (2018) Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146
13. Wang Y, Xu W (2018) Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud. Decis Support Syst 105:87–95
14. Chatterjee A, Gupta U, Chinnakotla MK, Srikanth R, Galley M, Agrawal P (2019) Understanding emotions in text using deep learning and big data. Comput Hum Behav 93:309–317
15. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
16. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems 25 (NIPS 2012)
17. Habibi Aghdam H, Jahani Heravi E (2017) Convolutional neural networks. In: Guide to convolutional neural networks: a practical application to traffic-sign detection and classification. Springer, Cham, pp 85–130
18. Govindan V, Shivaprasad A (1990) Character recognition—a review. Pattern Recognit 23:671–683
19. Hamid A, Haraty R (2001) A neuro-heuristic approach for segmenting handwritten Arabic text. In: Proceedings ACS/IEEE international conference on computer systems and applications, 2001, pp 110–113
20. Pal U, Chaudhuri B (2004) Indian script character recognition: a survey. Pattern Recognit 37:1887–1899
21. Biadsy F, Saabni R, El-Sana J (2011) Segmentation-free online Arabic handwriting recognition. Int J Pattern Recognit Artif Intell 25:1009–1033
22. Tappert CC, Suen CY, Wakahara T (1990) The state of the art in online handwriting recognition. IEEE Trans Pattern Anal Mach Intell 12:787–808
23. Plamondon R, Srihari SN (2000) Online and off-line handwriting recognition: a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 22:63–84
24. El-Sawy A, Loey M, Hazem E (2017) Arabic handwritten characters recognition using convolutional neural network. WSEAS Trans Comput Res 5:11–19
25. Younis KS (2017) Arabic handwritten character recognition based on deep convolutional neural networks. Jordan J Comput Inf Technol 3:2017
26. El-Melegy M, Abdelbaset A, Abdel-Hakim A, El-Sayed G (2019) Recognition of Arabic handwritten literal amounts using deep convolutional neural networks, Cham, pp 169–176
27. Torki M, Husseiny ME, Elsallamy A, Fayyaz M, Yaser S (2014) Window-based descriptors for Arabic handwritten alphabet recognition: a comparative study on a novel dataset. arXiv preprint arXiv:1411.3519
28. Loey M (31-08-2019) Arabic handwritten characters dataset. https://www.kaggle.com/mloey1/ahcd1
29. Alamri H, Sadri J, Suen CY, Nobile N (2008) A novel comprehensive database for Arabic off-line handwriting recognition. In: Proceedings of 11th international conference on frontiers in handwriting recognition, ICFHR, 2008, pp 664–669
30. Eikvil L (1993) OCR-optical character recognition. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.3684
31. Lensu A (2002) Computationally intelligent methods for qualitative data analysis. No. 23. University of Jyväskylä. https://www.semanticscholar.org/paper/Anssi-Lensu-Computationally-Intelligent-Methods-for-Lensu-Olsbo/f58234bfae6de53aa39110ed69f3438c59cb0304
32. Vadwala MA, Suthar MK, Karmakar MY, Thakkar N (2017) Survey paper on different speech recognition algorithm: challenges and techniques. Int J Comput Appl 175(1):31–36
33. Lawgali A (2015) A survey on Arabic character recognition. https://doi.org/10.14257/ijsip.2015.8.2.37
34. Tanner MA, Wong WH (1987) The calculation of posterior distributions by data augmentation. J Am Stat Assoc 82:528–540
35. Frühwirth-Schnatter S (1994) Data augmentation and dynamic linear models. J Time Ser Anal 15:183–202
36. Hamida S, Cherradi B, Ouajji H, Raihani A (2020) Convolutional neural network architecture for offline handwritten characters recognition. In: International conference Europe Middle East & North Africa information systems and technologies to support learning. Springer, Cham, pp 368–377. https://doi.org/10.1007/978-3-030-36778-7_41
37. Neri CG, Villegas OOV, Sánchez VGC, Nandayapa M, Azuela JHS (2020) A convolutional neural network for handwritten digit recognition. Int J Comb Optim Probl Inform 11:97–105
38. Clevert D-A, Unterthiner T, Hochreiter S (2015) Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289
39. Younis K (2018) Arabic handwritten character recognition based on deep convolutional neural networks. Jordanian J Comput Inform Technol 3(3)
40. Deng L (2012) The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process Mag 29:141–142
41. Torrey L, Shavlik J (2010) Transfer learning. In: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI Global, 2010, pp 242–264
42. Pan SJ (2009) Q. J. I. T. o. k. Yang, and d. engineering, A survey on transfer learning, vol 22, pp 1345–1359
43. Kim Y (2014) Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882
44. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958
45. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167
46. Hidaka A, Kurita T (2017) Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks. In: Proceedings of the ISCIE international symposium on stochastic systems theory and its applications, vol 2017. The ISCIE symposium on stochastic systems theory and its applications 2017
47. Undrestanding Convolutional Layers in Convolutional Neural Networks (CNNs). http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html
48. Mallick S, Nayak S (2018, May 22) Number of parameters and tensor sizes in a convolutional neural network (CNN). https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/
49. van Laarhoven T (2017) L2 regularization versus batch and weight normalization. arXiv preprint arXiv:1706.05350
50. Hara K, Saito D, Shouno H (2015) Analysis of function of rectified linear unit used in deep learning. In: 2015 international joint conference on neural networks (IJCNN), pp 1–8
51. Dunne RA, Campbell NA (1997) On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In: Proceedings of 8th Australian conference on neural networks, Melbourne, 1997, p 185

52. Koturwar S, Merchant S (2017) Weight initialization of deep neural networks (DNNS) using data statistics. arXiv preprint arXiv:1710.10570
53. Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: International conference on machine learning, 2013, pp 1139–1147
54. LeCun YA, Bottou L, Orr GB, Müller K-R (2012) Efficient backprop. In: Neural networks: tricks of the trade. Springer, pp 9–48
55. Klambauer G, Unterthiner T, Mayr A, Hochreiter S (2017) Self-normalizing neural networks. In: Advances in neural information processing systems, pp 971–980
56. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp 249–256
57. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, 2015, pp 1026–1034
58. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
59. Reddi SJ, Kale S, Kumar S (2019) On the convergence of adam and beyond. arXiv preprint arXiv:1904.09237
60. Zeiler MD (2012) ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701
61. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:2121–2159
62. Bottou L (2012) Stochastic gradient descent tricks. In: Neural networks: tricks of the trade. Springer, pp 421–436
63. Gulli A, Pal S (2017) Deep learning with Keras. Packt Publishing Ltd, Birmingham
64. Bisong E (2019) Google colaboratory. In: Building machine learning and deep learning models on Google cloud platform. Springer, pp 59–64
65. Joshi R (2016, September 9) Accuracy, precision, recall & F1 score: interpretation of performance measures. https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/
66. Accuracy, Precision, Recall and F1 Scores for Lawyers. (October 10, 2019). https://lawtomated.com/accuracy-precision-recall-and-f1-scores-for-lawyers/
67. Nicholson C. Evaluation metrics for machine learning—accuracy, precision, recall, and F1 defined. https://pathmind.com/wiki/accuracy-precision-recall-f1
68. Chase Lipton Z, Elkan C, Narayanaswamy B (2014) Thresholding classifiers to maximize F1 score. arXiv preprint arXiv:1402.1892
69. Powers DM (2011) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. https://dspace2.flinders.edu.au/xmlui/handle/2328/27165
70. Goutte C, Gaussier E (2005) A probabilistic interpretation of precision, recall and F-score, with implication for evaluation, vol 3408
71. Reed R, Marks RJ II (1999) Neural smithing: supervised learning in feedforward artificial neural networks. MIT Press, Cambridge
72. Al-Ayyoub M, Nuseir A, Alsmearat K, Jararweh Y, Gupta B (2018) Deep learning for Arabic NLP: a survey. J Comput Sci 26:522–531
73. Abdelazeem S, El-Sherif E. The Arabic handwritten digits databases: ADBase & MADBase. http://datacenter.aucegypt.edu/shazeem/
74. Alani A (2017) Arabic handwritten digit recognition based on restricted boltzmann machine and convolutional neural networks. Information 8:142
75. Ashiquzzaman A, Tushar AK, Rahman A (2017) Applying data augmentation to handwritten arabic numeral recognition using deep learning neural networks, arXiv preprint arXiv:1708.05969