Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.052016

ARTICLE

Optimised CNN Architectures for Handwritten Arabic Character Recognition

Salah Alghyaline*

Department of Computer Science, The World Islamic Sciences and Education University, Amman, 1101-11947, Jordan

*Corresponding Author: Salah Alghyaline. Email: salah.alghyaline@wise.edu.jo

Received: 20 March 2024 Accepted: 07 May 2024 Published: 20 June 2024

ABSTRACT

Handwritten character recognition is considered challenging compared with machine-printed characters due to the different human writing styles. Arabic is morphologically rich, and its characters have a high similarity. The Arabic language includes 28 characters. Each character has up to four shapes according to its location in the word (at the beginning, middle, end, and isolated). This paper proposed 12 CNN architectures for recognizing handwritten Arabic characters. The proposed architectures were derived from the popular CNN architectures, such as VGG, ResNet, and Inception, to make them applicable to recognizing character-size images. The experimental results on three well-known datasets showed that the proposed architectures significantly enhanced the recognition rate compared to the baseline models. The experiments showed that data augmentation improved the models' accuracies on all tested datasets. The proposed model outperformed most of the existing approaches. The best achieved results were 93.05%, 98.30%, and 96.88% on the HIJJA, AHCD, and AIA9K datasets.

KEYWORDS

Optical character recognition (OCR); handwritten arabic characters; deep learning

1 Introduction

Optical Character Recognition (OCR) converts printed or handwritten text inside the image into a text [1]. OCR has many applications such as form processing [2,3], automatic indexing [4,5] bank checks processing [6,7], card scanner [8,9], and text translation [10,11]. The extracted text is usually used for further processing according to the user's needs.

Recognizing Arabic characters is challenging due to many facts [12,13]; the Arabic language is morphologically rich. Also, each letter of the alphabet has many shapes according to its appearance in the word (at the beginning, middle, end, and isolated). Diacritics are also used in Arabic and have different locations (under and above the letter). Dots are essential to the letter structure and can be above and under the letter. Many Arabic letters have the same shape, but the position of the dots changes the meaning of the letter and therefore the meaning of the word. The Arabic alphabet is written in cursive lines. Therefore, recognizing handwritten characters is more challenging than printed text due to the different written styles used by different persons [12]. Character recognition is a fundamental stage for higher-level task, word, or sentence recognition. Many models were proposed for character-level recognition, some of them are reviewed in the related work section of this paper.

Enhancing character level recognition significantly improves the overall performance of the OCR system. Therefore, developing accurate and fast models for character-level recognition is crucial for researching the OCR domain.

Deep learning achieved remarkable results in image processing in terms of accuracy and speed [13–15]. Many Convolutional Neural Networks (CNN) architectures have been proposed for image classification and object detection. CNN consists of layers varying in number, type, and passed parameters. CNN layers include convolution, pooling, fully connected, drop, activation, shortcut, and others. In addition to the CNN architectures, optimization techniques, such as Adam and RMSprop techniques, are used to decrease the loss function and update the network weights. Many CNN architectures were proposed in the literature and are widely used in many applications in our lives. Some of the famous architectures are: VGG [16], GoogleNet [17], ResNet [18], EfficientNet [19], Vision Transformers (ViT) [20], MobileNet [21], Inception [22], AlexNet [23], and LeNet [24]. Increasing the number of layers does not constantly improve the recognition accuracy. The dimensions for written Arabic characters are small (usually 32×32 pixels), which requires special CNN architecture to avoid dimension and gradient vanishing after a certain number of layers of convolutions and pooling.

This paper investigates the performance of eight well-known CNN architectures on handwritten Arabic optical character recognition. Moreover, this paper modified some famous CNN architectures to improve the recognition rate. The eight used architectures are VGG, GoogleNet, ResNet, EfficientNet, Vision Transformers (ViT), MobileNet, Inception, and Shi et al. [25]. The experimental results on three well-known handwritten Arabic datasets, mainly HIJJA, AHCD, and AIA9K, proved that the proposed modification on these CNN architectures improved the optical character recognition rates. Experimental results demonstrated that data augmentation significantly improves the accuracy of OCR. The newly proposed models are optimized compared with the baseline models for the following reasons: They reduced the time complexity by decreasing the number of layers and parameters, addressed the overfitting problem, generalized well with unseen data, and improved the recognition rates.

The paper is organized as follows: The related work section reviews the Arabic language and the existing handwritten Arabic character recognition models. The materials and methods section shows the existing CNN architectures and the proposed CNN architectures. The experiment materials section explains the machine used to implement the code and the datasets used. The results and discussion section presents the accuracy of different proposed CNN Architectures and compares them with the existing approaches. The conclusion and future work are presented in the last section.

2 Related Work

2.1 The Arabic Language

Arabic is the formal language for 22 Arabic countries, with a population of more than 464 million in the year 2022 [26]. The Holy Quran is written in Arabic, and Muslims worldwide recite it in Arabic. Arabic is written from right to left in a cursive way. Arabic includes 28 characters, as shown in Table 1. Each character has up to four shapes which depend on the character's position in the word (at the beginning, middle, end, and isolated), as shown in Table 2. A sample of printed and handwritten Arabic characters is shown in Table 3. It is clear from the table that there are many similarities between some Arabic characters. These similarities make recognising handwritten Arabic characters difficult and increase the recognition error rate [27,28]. Moreover, the handwritten character could suffer from rotation, and it is not identical to printed characters due to the different writing styles that people use. This makes it confusing for the CNN model to recognize them accurately compared with the printed

characters. Many studies have shown that Arabic script is more challenging in terms of OCR compared to Latin script, and most of the existing OCR tools are made for the English language [1,29,30].

Table 1: The printed Arabic alphabet with their names in English

ا	Alif	ج	Jeem	ذ	Dhaal	ش	Sheen	ظ	Zaa	ق	Qaaf	ن	Noon
ب	Baa	ح	Haa	ر	Raa	ص	Saad	ع	‘Ayn	ك	Kaaf	ه	Haa’
ت	Taa	خ	Khaa	ز	Zaay	ض	Daad	غ	Ghayn	ال	Laam	و	Waaw
ث	Thaa	د	Daal	س	Seen	ط	Taa’	ف	Faa	م	Meem	ي	Yaa

Table 2: Sample of four handwritten Arabic characters at different positions in the word

Character	Beginning	Middle	End	Isolated
Kaaf				
Taa				
Meem				
Yaa				

Table 3: Sample of eight printed and handwritten Arabic characters

Printed Arabic character	ب	ت	ث	ي	م	د	ا	س
Handwritten Arabic character								
Character pronunciation	Baa	Taa	Thaa	Yaa	Meem	Daal	Alif	Seen

2.2 Handwritten Arabic Recognition

The section reviews the related work in handwritten Arabic recognition. The input for the OCR system could be machine-printed or human handwritten characters; the handwritten OCR is more challenging and still an unsolved problem due to the different handwritten styles and formats that the language users use [31].

Reference [32] built a CNN architecture of two Conv2d layers and two max pooling layers. The Conv2d filters are 80 and 64 for the first and the second convolutional layers, respectively. Filter size is 7×7 for the Conv2d filter and 2×2 for the max pooling layers. Two fully connected layers were used; their sizes are 1024 and 28, respectively. SoftMax is used to classify the network output into 28 characters. A new dataset called AHCD was proposed, and the proposed architecture achieved 94.9% accuracy on the AHCD dataset.

Reference [33] designed a CNN with three Conv2d and kernel size is 7×7, each followed by a max pooling layer; the network flattens and passes to a fully connected layer with size 1024. SVM was used

to classify the resulting vector into 28 characters. The system achieved 95.07% accuracy on a private dataset of 840 images.

Reference [34] built a CNN to recognize handwritten Arabic characters. The architecture includes two Conv2d layers, each with 32 filters, and the filter size is 5 × 5. Each Conv2d layer is followed by a max pooling layer with 2 × 2. The fifth layer is a Conv2d layer with 64 filters and a 4 × 4 filter size. The output is passed to a fully connected layer with 256 sizes and another fully connected layer with 28 sizes. Finally, it was classified into 28 characters using SoftMax. The experiments on three datasets, AHCR, AHCD, and HIJJA achieved 89.9%, 95.4% and 92.5%, respectively.

Reference [35] proposed 14 different CNN architectures for recognizing handwritten Arabic text. All architectures have three Conv2d and three max pooling layers and from two to three fully connected layers. However, the architectures used different filter numbers and sizes and tested different activation functions. The HMBD dataset was used to evaluate the model. Data augmentation with the HMBD dataset was used to boost the recognition accuracy and the best reported accuracy was 92.88%.

The CNN architecture comprises four Conv2d layers, a max pooling layer after each two Conv2d layers [36]. The filter numbers are 16 for the first and second Conv2d layers and 32 for the third and fourth layers. The filter size is 3 × 3 for Conv2d layers and 2 × 2 for the max pooling layers. There are two fully connected layers, each with a length of 100. The architecture used a dropout layer of 0.6 after each fully connected layer. The experiments performed on AHCD and HIJJA datasets showed that the architectures achieved recognition rates of 98.48% and 91.24%.

The proposed model includes three Conv2d layers, each containing 32 filters, followed by max pooling and dropout layers [37]. Then, there are two Conv2d layers, each consisting of 128 filters, followed by max pooling and dropout layers. The output is passed into two fully connected layers, each with a size of 512. Then, the output is classified using the SoftMax classifier. The experiments on two datasets, AHCD and HIJJA achieved 98%, and 91% accuracy, respectively.

VGG16 is used to classify handwritten Arabic characters. Different activation and optimization functions were tested with the AHCD dataset. The best-achieved accuracy by the model was 95.89 [38].

The model is based on handcrafted features such as projections, profiles, widths and heights, extrema, concave arcs, endpoints, holes, and junctions [39]. The extracted features were then classified using KNN, SVM, and RF. Experiments on the HCDB and AIA9K datasets showed that the architectures achieved recognition rates of 97.97% and 92.91%, respectively.

The model is based on ResNet50. It replaced the architecture's last layer with Random Forest (RF) and Support Vector Machine (SVM) classifiers [40]. It is reported that this reduces the classification time and increases the recognition rate. The reported accuracies were 92.4%, 99%, and 95% for HIJJA, AHCD, AIA9K, and datasets.

Reference [41] proposed an approach to segment and recognize handwritten Arabic characters. The approach segments the word into sub-words and then segments the sub-words into characters. The Alex Net architecture was used to segment and identify the Arabic characters. The IESK-ArDB Arabic dataset was used to evaluate the models. They achieved 96% for the word-to-sub-word segmentation accuracy and 96.97% for the word-to-character segmentation accuracy.

3 Materials and Methods

3.1 VGG Model

VGG [16] stands for Visual Geometry Group, the group at Oxford University that proposed the architecture. The team reported that VGG16 and VGG19 achieved the state of the art on the ImageNet dataset. The VGG16 architecture includes 13 Cov2D Layers, 5 MaxPool2D layers, one Flatten Layer, and three fully connected layers. VGG19 has the same architecture as VGG16 but has three more Cov2D layers. The first two columns in Table 4 show the architectures for VGG19 and VGG16, respectively. The Cov2D filter size is a window of 3 × 3, whereas the MaxPool2D window size is 2 × 2. ReLU (Rectified Linear Unit) is activated after each Cov2D layer.

Table 4: Proposed VGG-based CNN architectures for Arabic handwritten recognition

ConvNet configuration							
VGG19	VGG16	VGGA	VGGB	VGGC	VGGD	VGGE	VGGF
25 layers	22 layers	18 layers	17 layers	20 layers	23 layers	19s layers	25 layers
Input image (32 × 32 grayscale image)							
Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64
Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64	Conv2D-64
Maxpool							
					Drop 0.5		Drop 0.5
Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128
Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128	Conv2D-128
Maxpool							
					Drop 0.5		Drop 0.5
Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256
Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256
Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256	Conv2D-256
Conv2D-256				Conv2D-256			Conv2D-256
Maxpool							
					Drop 0.5		Drop 0.5
Conv2D-512	Conv2D-256	Conv2D-256	Conv2D-512	Conv2D-512	Conv2D-256	Conv2D-256	Conv2D-512
Conv2D-512	Conv2D-256	Conv2D-256	Conv2D-512	Conv2D-512	Conv2D-256	Conv2D-256	Conv2D-512
Conv2D-512	Conv2D-256	Conv2D-256		Conv2D-512	Conv2D-256	Conv2D-256	Conv2D-512
Conv2D-512				Conv2D-512			Conv2D-512
Maxpool							
					Drop 0.5		Drop 0.5
Conv2D-512	Conv2D-512						
Conv2D-512	Conv2D-512						
Conv2D-512	Conv2D-512						

(Continued)

Table 4 (continued)

ConvNet configuration							
VGG19	VGG16	VGGA	VGGB	VGGC	VGGD	VGGE	VGGF
25 layers	22 layers	18 layers	17 layers	20 layers	23 layers	19s layers	25 layers
Conv2D-512 Maxpool	Maxpool						
			Flatten FC-4096 FC-4096				
					Drop 0.5	Drop 0.5	Drop 0.5
			FC-number_of_classes SoftMax				

The VGG19 and VGG16 were designed for large-scale images. The input is an RGB image with 224 × 224 dimensions. However, grayscale images with 32 × 32 dimensions are used in most datasets for handwritten Arabic characters. Therefore, this paper proposed six new VGG-based architectures: VGGA, VGGB, VGGC, VGGD, VGGE, and VGGF to enhance the recognition rate for handwritten Arabic OCR. Table 4 shows the eight architectures; each column represents a single architecture. The new architectures differ in the number of Cov2D layers, number of filters, and dropout layers. The architecture layers ranged from 17 in VGGB to 25 in VGGF, whereas the Conv2D ranged from 9 in VGGB to 12 in VGGF. Drop layers with 50% are used after each MaxPool2D in architectures VGGD and VGGF. VGGE has a single drop layer before the last fully connected layer. The numbers of filters used were 64, 128,256, and 512.

The VGG19 and VGG16 architectures include four blocks of Cov2D Layers, each followed by a MaxPool2D layer; the MaxPool2D layer reduces the spatial dimensions of the input image by 50%. Therefore, the architectures start with an image of 32 × 32 pixels and 64 filters (32 × 32 × 64) and end with 1 × 1 pixels and 512 filters (1 × 1 × 512). Reducing the spatial dimensions to 1 × 1 pixels might result in loss of spatial information. The proposed VGGA to VGGF removed the last block of Cov2D Layers. The same input as the VGG19 and VGG16, but the output is 2 × 2 × 512, improving the network's ability to learn more. Moreover, adding drop layers with 50% prevents overfitting and allows the network to learn more robust features. It also does not make it rely on specific neurons during training.

3.2 Inception Model

The inception model [22] includes a sequence of Cov2D, Batch Normalization, and MaxPool2D layers. Different Cov2D respective fields are used (1 × 1, 3 × 3, 5 × 5). The Inception Modules concatenate the outputs from different convolutional layers and capture features from different scales. Fig. 1 shows how Inception Modules of type A concatenate features from four previous convolutional layers [22]. InceptionV3 is a very deep architecture with 159 layers. The input for InceptionV3 is an RGB image with 299 × 299 × 3 dimensions. InceptionV3 does not accept less than 75 × 75 × 3 dimensions. However, a small grayscale image with 32 × 32 dimensions is used in most datasets for handwritten Arabic characters. Therefore, using a very deep model does not yield a good accuracy result. This paper proposes four new Incept-based architectures: Inception_A, Inception_B,

Inception_C, and Inception_D. Table 5 shows the four architectures, each column represents a single architecture. The proposed architectures use fewer layers than the InceptionV3 model. Three filters: 64,128, and 256 like the VGG model filters were used.

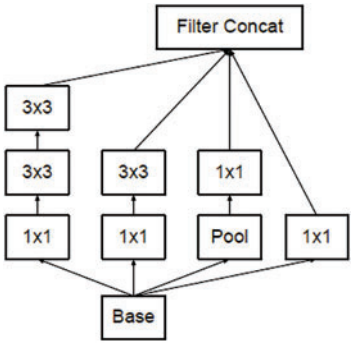


Figure 1: Inception module of type A

Table 5: Proposed Incept-based CNN architectures for handwritten Arabic recognition

Inception_A	Inception_B	Inception_C	Inception_D
24 layers	24 layers	27 layers	37 layers
	Input Image (32 × 32 grayscale image)		
Conv2D (32)	Conv2D (64)	Conv2D (64)	Conv2D (64)
	BatchNormalization		
Activation(ReLU)	Activation(ReLU)	Activation(ReLU)	Activation(ReLU)
Conv2D (32)	Conv2D (64)	Conv2D (64)	Conv2D (64)
	BatchNormalization		
Activation(ReLU)	Activation(ReLU)	Activation(ReLU)	Activation(ReLU)
Conv2D (64)	Conv2D (64)	Conv2D (64)	Conv2D (64)
	BatchNormalization		
Activation(ReLU)	Activation(ReLU)	Activation(ReLU)	Activation(ReLU)
MaxPooling2D	MaxPooling2D		MaxPooling2D
Conv2D (80)	Conv2D (128)	Conv2D (128)	Inception layer as shown in Fig. 1
	BatchNormalization		
Activation(ReLU)	Activation(ReLU)	Activation(ReLU)	
Conv2D (192)	Conv2D (128)	Conv2D (128)	
	BatchNormalization		
Activation(ReLU)	Activation(ReLU)	Activation(ReLU)	
Conv2D (64)	Conv2D (256)	Conv2D (256)	
	BatchNormalization		
Activation(ReLU)	Activation(ReLU)	Activation(ReLU)	
	MaxPooling2D		
		Conv2D (256)	
		BatchNormalization	
		Activation(ReLU)	

(Continued)

Table 5 (continued)

Inception_A	Inception_B	Inception_C	Inception_D
		Flatten	
		FC-128	
		FC number_of_classes	
		soft-max	

This paper investigated the performance of InceptionV3 architecture on handwritten Arabic character recognition and proposed new models based on InceptionV3. Table 6 shows the Inception_D model. It includes three Cov2D layers, each Cov2D followed by a BatchNormalization and a ReLU activation layers. After that, an Inception module of type A is added to the architecture. Then, the output is flattened and passed to two fully connected layers. Finally, SoftMax is used for classification. The number of layers in Inception_D is 37 (including batch normalization, activation, flatten, and fully connected layers) and the number of parameters is 9.1 M. Inception_A includes the first 24 layers of the InceptionV3 model without using the Inception module. In Inception_B, the architecture of Inception_A is used, but the number of filters is changed to 64, 128, and 256 like the VGG model. In Inception_C, three more layers were added compared to Inception_B: Cov2D, BatchNormalization, and ReLU

Table 6: Proposed Inception_D CNN architectures

Type of layer	Patch size	Input size
Conv2D (64)	(3 × 3)	32 × 32 × 1
Conv2D (64)	(3 × 3)	32 × 32 × 64
Conv2D (64)	(3 × 3)	30 × 30 × 64
MaxPooling2D	(2 × 2)	28 × 28 × 64
Inception	As in Fig. 1	14 × 14 × 64
Flatten	–	14 × 14 × 352
FC-128	–	1 × 68992
FC Number_of_classes	logits	1 × 128
SoftMax	classifier	1 × Number_of_classes

The number of layers is reduced from 159 in InceptionV3 to 24, 27, and 37 layers in the proposed architectures. Additionally, the number of filters is changed to become like VGG architecture. The InceptionV3 architecture is deep. Deep networks are more prone to overfitting mainly when applied to character recognition with small input image sizes (32 × 32 pixels). InceptionV3 is designed for large images with high resolution and rich features. The small pictures have fewer details making it hard for the model to identify the character's label correctly. The proposed architectures aim to solve the problem of overfitting by simplifying the InceptionV3 model complexity. Also, it aims to reduce its ability to memorize the training data and perform well in unseen data.

3.3 ResNet Model

ResNet [18] achieved first place in the ILSVRC 2015 competition. ResNet tried to address the vanishing gradient problem using the residual learning technique. Fig. 2 shows the structure of the residual learning block as presented in [18]. This allows the network to learn even with deeper networks. Instead of stacking layers above each other (such as in VGG architecture), ResNet uses a skip (shortcut) layer. The skip layer ignores a certain number of layers and adds the original input to the output of the Conv layer. This paper evaluated the performance of the original Resnet50 (48 Conv2D, one Avg Pool, and one fully connected layer) which includes 177 layers (counting all layers from the input to the output layer not only Conv layers). Two additional ResNet-based models were proposed, mainly ResNet_A and ResNet_B. These two models have 20 and 29 layers, respectively, much smaller than Resnet50 (177 layers). The proposed ResNet_A architecture includes 5 Conv2D layers. Each Conv2D layer is followed by batch normalization and activation layers. As shown in Fig. 3, there are two residual operations. The output of the third Conv2D is added to the output of the first Conv2D layer, and the new features are passed to the next layer. The second residual operation includes adding the output of the third Conv2D with the output of the fifth Conv2D. The filter size is 3×3 and the number of filters is 32 for all Conv2D layers. Compared with ResNet_A, the ResNet_B architecture includes two residual blocks but with two more Conv2D layers, adding Conv2D for each residual block, as shown in Fig. 4.

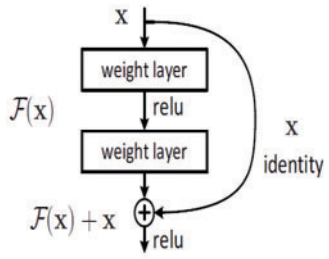


Figure 2: Residual learning block

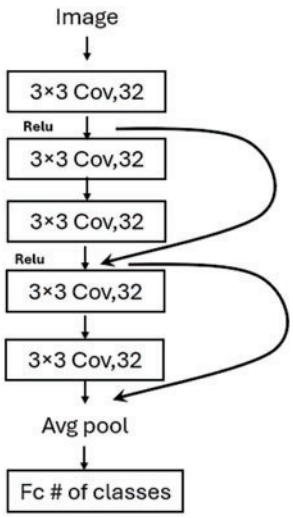


Figure 3: Proposed ResNet_A

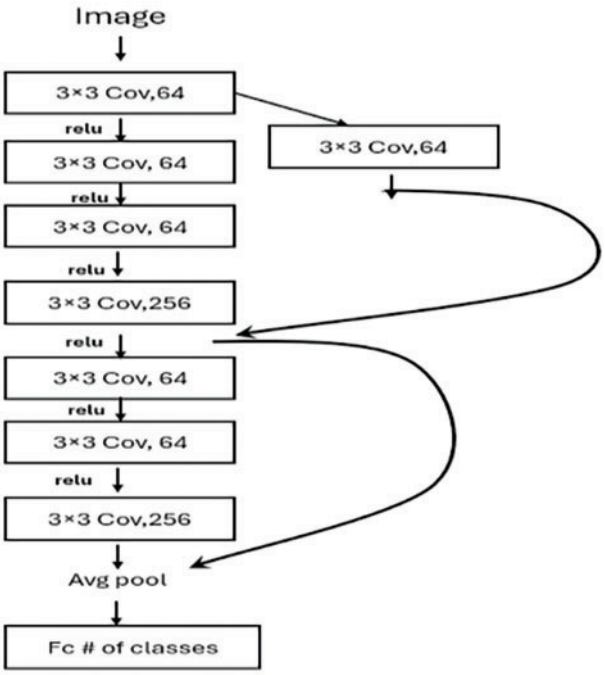


Figure 4: Proposed ResNet_B

ResNet was designed for large RGB images with 224 × 224 pixels. The large number of layers and parameters leads to overfitting when applied to a small image of Arabic characters. Therefore, the proposed architectures reduced the number of layers. This makes computation more efficient, and the model generalizes well with unseen data.

3.4 GoogLeNet Model

GoogLeNet architecture [17] won the 2014 ILSVRC competition. It used inception modules to perform multiple convolutions simultaneously and then combine the results from these convolutional layers. Therefore, GoogLeNet combines image features at different scales. Performing multiple convolutions simultaneously reduces the running time and improves accuracy.

3.5 EfficientNet Model

Reference [19] proposed eight architectures from EfficientNet-B0 to EfficientNet-B7. The authors reported that EfficientNet-B7 achieved the state of the art results on CIFAR-100 and ImageNet datasets compared with other architectures such as ResNet, DenseNet, Inception, Xception, ResNeXt, PolyNet, AmoebaNet, PNASNet, and GPipe. The architecture is smaller and faster than the best existing architectures. The EfficientNet used MobileNets and ResNet architectures as base models and scaled them from three sides: Width, depth, and resolution in a balanced way.

3.6 Vision Transformer (ViT) Model

Reference [20] introduced a transformer encoder to the CNN architectures. The input image is divided into patches and converted into a vector. The position of the patch is embedded in the vector. Then, the vector is passed into a transformer encoder. ViT achieved remarkable results when it was

trained on a large scale dataset. The model achieved the state of the art results compared to other CNN architectures.

3.7 Shi et al.'S Model

Shi et al. [25] proposed end-to-end OCR for natural scenes. They reported that the architecture achieved superior results compared to prior arts in the three datasets: IIIT-5K, ICDAR, and Street View Text. The architecture used a sequence of convolutional layers and recurrent neural networks (bidirectional LSTM) (19 layers). The proposed architecture showed superior results on IC03, IC13, IIIT5k, and SVT datasets.

3.8 MobileNet Model

Reference [21] was designed for mobile phones and small embedded system applications. Therefore, its goal is to achieve high accuracy with less computational cost to make it applicable to small devices. It is reported that MobileNet has accuracy close to that of the VGG16 model and is 32 times faster than the VGG16 model.

4 Experiment Materials

The experiments were implemented using Python and TensorFlow [42]. The machine that is used to run the code is an Intel(R) Core (TM) i5-8600K CPU with 3.60 GHz speed, 40 GB RAM, and GTX1080 GPU. Three well-known datasets for handwritten Arabic character recognition were used to evaluate the performance of the different CNN architectures. HIJJA, AHCD, and AIA9K, respectively, according to their size. HIJJA is the biggest and includes four shapes for each character (at the beginning, middle, end, and isolated). Table 7 shows the hyperparameters used to evaluate all models.

Table 7: Hyper-parameters for experiments

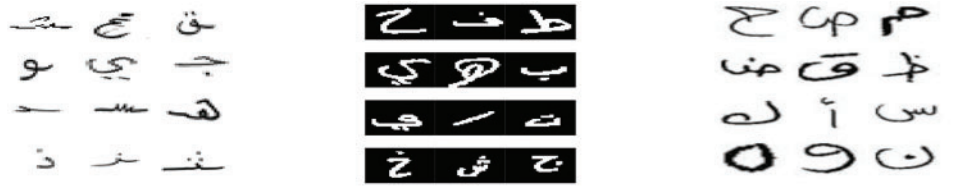
Hyperparameters	Setting
Input size	32 × 32 (Grayscale image)
Batch size	256
Learning rate	0.001
Epochs	1000
Early stopping	Patience = 50 epochs for val_accuracy and restore_best_weights = True
Optimizer	Adam
Loss function	Categorical_Crossentropy

4.1 Dataset Details

4.1.1 HIJJA Dataset

HIJJA [43] includes 47434 images. Each image is 32 × 35 pixels. The images represent 29 Arabic characters (28 Arabic alphabets and "hamza" character) written by 598 Arabic native speakers in Riyadh, Saudi Arabia in 2019. Students' ages ranged from 7–12 years old. The dataset consists of 29 folders. Each folder includes a subfolder representing a single Arabic character. The subfolder consists

of different forms of writing a single character at the beginning, middle, and end of the word. These various forms of writing Arabic characters make the dataset more complicated than other datasets that use a single form of character writing (isolated form). A sample of 12 pictures from the HIJJA dataset is shown in Fig. 5A. The dataset used in the experiments was obtained from https://github.com/israksu/Hijja2.



(A) The HIJJA dataset (B) The AHCD dataset. (C) The AIA9K dataset.

Figure 5: Pictures from the used datasets

4.1.2 AHCD Dataset

AHCD [32] includes 16800 images of handwritten Arabic characters, and each image is 32 × 32 × 1 (Grayscale image). 13440 images are used for training and 3360 for testing. The Arabic characters were written by 60 people from Egypt and their ages were between 19–40. The images represent 28 Arabic alphabet characters; the isolated form of Arabic characters was used in the dataset. A sample of 12 pictures from the AHCD dataset is shown in Fig. 5B.

4.1.3 AIA9K Dataset

The AIA9K [44] contains 8737 pictures that represent 32 × 32 × 1 grayscale images. Each image includes a single isolated Arabic character. The pictures are divided into three categories: 6115, 1310, and 1312 for training, validating, and testing, respectively. The dataset was written by 107 students from Alexandria University in Egypt. The participants' ages were between 18 and 25 years old. A sample of 12 pictures from the AIA9K dataset is shown in Fig. 5C. The dataset used in the experiments was obtained from http://www.eng.alexu.edu.eg/staff/mehussein/public_html/AIA9k/index.html.

5 Results and Discussion

This section shows the performance of different proposed deep-learning architectures for recognizing handwritten Arabic characters. Eight architectures were used and modified to improve character recognition rates: VGG, GoogleNet, ResNet, EfficientNet, Vision Transformers (ViT), MobileNet, Inception, and Shi et al. [25]. The evaluation was performed on three popular datasets in the literature: HIJJA, AHCD, and AIA9K. The Accuracy, Recall, Precision, and F1-score metrics were used for the models' evaluation as shown in Eqs. 1–4 TP is a true positive, FN is a false negative, TN is a true negative, and FP is a false positive.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4}$$

5.1 VGG Model

Six architectures based on the VGG model were proposed and compared with the baseline models VGG16 and VGG19. The proposed architectures are VGGA, VGGB, VGGC, VGGD, VGGE, and VGGF. Table 8 shows that the VGGD architecture achieved the best overall accuracy of evaluated architectures on all tested datasets. The accuracies for VGGD architecture are 91.54%, 98.01%, and 95.88% for HIJJA, AHCD, and AIA9K, respectively. Table 8 shows the number of parameters (in millions) used in these architectures. VGGB used the smallest parameters (30.6 M), whereas VGG19 used the largest number of parameters (39 M).

Table 8: The accuracy of the proposed VGG-based CNNs on the three benchmark datasets

ConvNset config.	HIJJA (%)	AHCD (%)	AIA9K (%)	# of layers	# of parameters
VGG16	90	95.89	94.74	25	33.7 M
VGG19	90.32	96.88	93.67	22	39 M
VGGA	89.9	97.2	94.51	18	33.7 M
VGGB	90.2	97.3	93.98	**17**	**30.6 M**
VGGC	90.3	97.32	94.44	20	35.9 M
VGGD	**91.54**	**98.01**	**95.88**	23	32.9 M
VGGE	90	97.65	94.13	20	32.9 M
VGGF	91.0	97.83	94.97	25	35.9 M

5.2 Inception Model

Four architectures based on Inception architecture were proposed and compared with the baseline architecture of InceptionV3. The proposed architectures are Inception_A, Inception_B, Inception_C, and Inception_D. The baseline InceptionV3 model includes 159 layers and has several Inception modules. Table 9 shows that the InceptionV3 has the lowest accuracy among all tested architectures. In Inception_D, layers are reduced from 159 to 37 including only one Inception module. The accuracy in Inception_D improved significantly in the three tested datasets compared with the baseline model. However, in Inception_C, the number of layers was reduced to 27, the Inception modules were removed, and the filter sizes were modified. Inception_C outperformed all Inception-based architectures for all datasets. Both Inception_A and Inception_B have 24 layers; however, in Inception_B, the filterer's sizes were changed as VGG model filters, and this modification improved the accuracy as shown in Table 9. The table shows the number of parameters used by each model. InceptionV3 used 22.3 M, whereas the Inception_A model used only 0.4 M. Using the inception module significantly increased the number of parameters for the Inception_D Model (9.1 M).

5.3 ResNet Model

Two ResNet-based architectures were derived from the baseline model ResNet50. ResNet50 is a very deep network with 177 layers. Table 10 shows that the derived architecture ResNet_A and ResNet_B outperformed the baseline model, and the number of layers and parameters have been

reduced significantly. ResNet_B achieved the best accuracy for all test datasets, and the number of parameters is about 157 thousand, whereas the number of parameters for ResNet50 is 23.6 million.

Table 9: The accuracy of the proposed Incept-based CNNs on three benchmark datasets

ConvNset config.	HIJJA (%)	AHCD (%)	AIA9K (%)	# of layers	# of parameters
Inception V3	57.7	83.6	82.77	159	22.3 M
Inception_A	90.7	97.47	92.99	**24**	**0.4 M**
Inception_B	91.6	97.68	94.05%	**24**	0.89 M
Inception_C	**91.6**	**97.89**	**94.51**	27	1.2 M
Inception_D	86.85	95.15	92.3	37	9.1 M

Table 10: The accuracy of the proposed ResNet-based CNNs on three benchmark datasets

ConvNset config.	HIJJA (%)	AHCD (%)	AIA9K (%)	# of layers	# of parameters
ResNet50	85.7	94.76	91.39	177	23.6 M
ResNet_A	83.22	96.4	91.84	20	38,942
ResNet_B	**90.76**	**97.92**	**93.75**	29	157,086

5.4 Data Augmentation

This section evaluates the impact of data augmentation (generating new data) on the datasets. The augmentation process increases the dataset size to improve model accuracy during the training. Training models are time-consuming. The four models that achieved the best accuracy are evaluated after data augmentation: VGGD, Inception_B, Inception_C, and ResNet_B. The volume of the generated data is three times the volume of the original training data split. Table 11 shows the augmentation operations that were applied to the datasets. In the first row, generated images randomly zoom in up to 20% compared with the original images. The values for the augmentation operations mostly ranged from 0% to 20% representing a slight to moderate transformation in the original images without extreme distortion. These transformations make the model robust to image variations. The experiments showed that it is adequate to enhance the accuracy of character recognition.

Table 12 clearly shows that data augmentation improved the recognition rate for all models and datasets. The VGGD model achieved 91.54 accuracy on the HIJJA dataset. After data augmentation, the same model accuracy improved to 93.05%. The augmentation process improved the recognition rate for VGGD by 1.51%, 0.29%, and 1% on HIJJA, AHCD and AIA9K datasets, respectively. The accuracy of model Inception_B improved by 0.7%, 0.56%, and 1.75% on HIJJA, AHCD, and AIA9K datasets. Meanwhile, in Inception_C, the improvement was 0.91%, 0.32%, and 0.92% on the HIJJA, AHCD, and AIA9K datasets. In ResNet_B, the improvement was 1.18%, 0.26%, and 0.46% on the HIJJA, AHCD, and AIA9K datasets. The best improvement was using Inception_B on the AIA9K dataset with a 1.75% improvement. The smallest improvement was ResNet_B on the AHCD dataset with 0.26%.

Table 11: The augmentation operations that were applied to the three benchmarks

Operation	Value
Zoom	0%–20%
Shift images height	0%–20%
Shift images width	0%–20%
Shear transformations	0%–20%
Flip horizontally	True
Rotate	0–40 degrees
Fill in missing pixels after transformations.	'Nearest'

Table 12: The effect of data augmentation on accuracy

ConvNset config.	HIJJA (%)	AHCD (%)	AIA9K (%)
VGGD	91.54	98.01	95.88
VGGD + AGUM	**93.05**	**98.30**	**96.88**
Inception_B	91.60	97.68	94.05%
Inception_B + AGUM	**92.30**	**98.24**	**95.80**
Inception_C	91.60	97.89	94.51
Inception_C + AGUM	**92.51**	**98.21**	**95.43**
ResNet_B	90.76	97.92	93.75
ResNet_B + AGUM	**91.94**	**98.18**	**94.21**

5.5 Comparison with Existing Approaches

Deep learning models were used and modified in the previous sections to recognise handwritten Arabic characters. Table 13 summarises the performance of the proposed architectures in terms of accuracy with other deep learning architectures. It is clear from Table 13 that the proposed architectures ResNet_B + AGUM, Inception_B + AGUM, and VGGD + AGUM outperformed the baseline models and other known CNN architectures like GoogLeNet, EfficientNet, ViT, Shi et al., and MobileNetV2. VGGD + AGUM achieved the highest accuracy for the three datasets. Table 14 presents the F1-score and precision metrics for the best-proposed architectures on three datasets. The result is consistent with Table 13, which shows that VGGD + AGUM achieved the best results followed by Inception_B + AGUM and ResNet_B + AGUM, respectively.

Table 13: The accuracy of handwritten Arabic recognition rates on different deep-learning models

Model	HIJJA	AHCD	AIA9K	# of layers	# of parameters
GoogLeNet	89.6	97.23	93.25	25	8.4 M
EfficientNet-B0	87.4	97.29	94.28	237	4.0 M
EfficientNet-B7	90.3	97.71	95.96	813	64 M

(Continued)

Table 13 (continued)

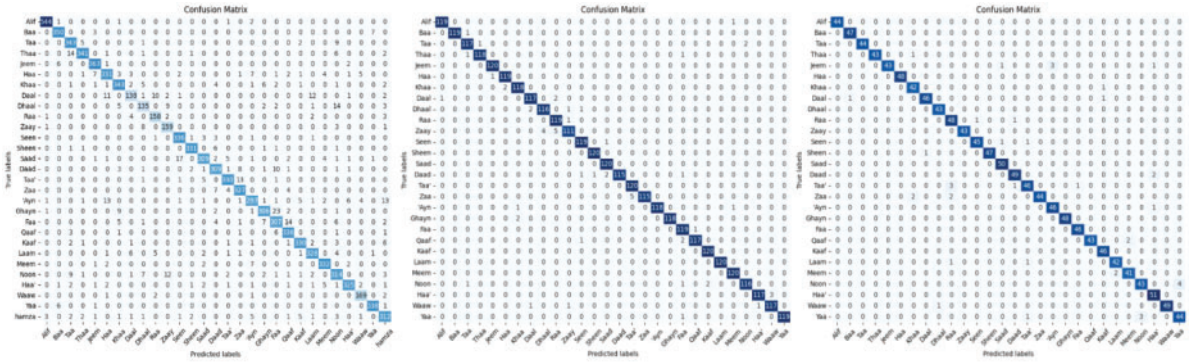
Model	HIJJA	AHCD	AIA9K	# of layers	# of parameters
Vision transformer (ViT)	53.7	67.86	64.02	34	0.3 M
Shi et al. [25]	87.6	96.58	94.21	19	8.7 M
MobileNetV2	82.8	95.42	75.61	158	2.3 M
VGG19	90.32	96.88	93.67	22	39 M
ResNet50	85.7	94.76	91.39	177	23.6 M
InceptionV3	57.7	83.6	82.77	159	22.3 M
ResNet_B + AGUM	91.94	98.18	94.21	29	157,086
Inception_B + AGUM	92.3	98.24	95.80	24	0.89 M
VGGD + AGUM	**93.05**	**98.30**	**96.88**	23	32.9 M

Table 14: The F1-score and precision metrics results for the best-proposed models

ConvNset config.	HIJJA		AHCD		AIA9K	
	F1	Precision	F1	Precision	F1	Precision
ResNet_B + AGUM	91.93	91.94	98.18	98.21	94.18	94.4
Inception_B + AGUM	92.34	92.37	98.25	98.29	95.80	95.93
VGGD + AGUM	93.04	93.08	98.30	98.33	96.87	96.95

Fig. 6 shows the VGGD model's confusion matrix with data augmentation on three datasets. AHCD and AIA9K datasets have almost the same number of samples for each character. HIJJA is larger than the previous dataset and some classes have fewer samples than others such as Daal('د'), Dhaal('ذ'), Raa('ر'), Zaay('ز'), and Waaw ('و'). It is clear from the diagonal line that the model predicted most of the characters correctly. Due to the similarities between Arabic characters, there is an error in predicting some of the characters. For example, in the HIJJA dataset, ten pictures of Daal character ('د') were predicted as Raa ('ر'), 12 Daal ('د') were predicted as Laam ('ل'),14 Dhaal ('ذ') were indicated as Noon ('ن'), 13 Ayn ('ع') were indicated as Hamza ('ء').

Table 15 compares the best accuracy achieved by the proposed VGGD + AGUM model in this paper and other approaches on HIJJA, AHCD, and AIA9K datasets. The proposed approach achieved the best results in HIJJA and AIA9K datasets. However, it achieved the second-best accuracy on the AHCD dataset.



(A) The HIJJA dataset (B) The AHCD dataset (C) The AIA9K dataset

Figure 6: Confusion matrix for VGGD + AGUM

Table 15: Comparison between the proposed VGGD + AGUM model and other approaches

AIA9K dataset			AHCD dataset			HIJJA dataset		
Method	Year	Accuracy	Method	Year	Accuracy	Method	Year	Accuracy
[43]	2021	88.0	[32]	2017	94.90	[45]	2017	94.8
[34]	2021	92.5	[45]	2017	97.60	[44]	2014	94.28
[36]	2022	91.2	[46]	2021	98.21	[47]	2021	93.30
[37]	2022	91.0	[34]	2021	95.4	[48]	2021	95.20
This paper	**2024**	**93.05**	[43]	2021	97.00	[39]	2022	92.91
			[49]	2022	96.78	[40]	2023	95.00
			[36]	2022	98.48	This paper	**2024**	**96.88**
			[37]	2022	98.00			
			[38]	2024	95.89			
			This paper	**2024**	**98.30**			

6 Conclusion and Future Work

Recognizing Arabic characters is more challenging compared with Latin characters. Arabic characters are highly similar; the same character has up to four shapes according to its position in the word. This paper proposed 12 CNN architectures for handwritten Arabic character recognition. The proposed architectures are based on well-known architectures in the literature such as VGG, ResNet, and Inception. The experimental results on HIJJA, AHCD, and AIA9K datasets showed that the proposed architectures achieved superior results compared to the existing Arabic handwritten character recognition approaches. The accuracy of the models can also be improved by using data augmentation. Future works include developing new CNN architectures and datasets for recognizing Arabic words and characters.

Acknowledgement: None.

Funding Statement: The author received no specific funding for this study.

Author Contributions: Study conception and design: Salah Alghyaline; analysis and interpretation of results: Salah Alghyaline; draft manuscript preparation: Salah Alghyaline. The author reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets adopted in the current study are publicly available online.

Conflicts of Interest: The author declares that he has no conflicts of interest to report regarding the present study.

References

[1] S. Faizullah, M. S. Ayub, S. Hussain, and M. A. Khan, "A Survey of OCR in Arabic language: Applications, techniques, and challenges," *Appl. Sci.*, vol. 13, no. 7, pp. 4584, 2023. doi: 10.3390/app13074584.

[2] P. Batra, N. Phalnikar, D. Kurmi, J. Tembhurne, P. Sahare and T. Diwan, "OCR-MRD: Performance analysis of different optical character recognition engines for medical report digitization," *Int. J. Inf. Tecnol.*, vol. 16, no. 1, pp. 447–455, 2024. doi: 10.1007/s41870-023-01610-2.

[3] A. A. Manjunath *et al.*, "Automated invoice data extraction using image processing," *IAES Int. J. Artif. Intell.*, vol. 12, no. 2, pp. 514, 2023. doi: 10.11591/ijai.v12.i2.pp514-521.

[4] H. Liu *et al.*, "Automated player identification and indexing using two-stage deep learning network," *Sci. Rep.*, vol. 13, no. 1, pp. 10036, Jun. 2023. doi: 10.1038/s41598-023-36657-5.

[5] M. Bulín, J. Švec, and P. Ircing, "The system for efficient indexing and search in the large archives of scanned historical documents," in *Eur. Conf. Info. Retrieval.*, Dublin, Ireland, 2023, pp. 206–210.

[6] P. Kunekar, K. Vayadande, O. Kulkarni, K. Ingale, R. Kadam and S. Inamdar, "OCR based cheque validation using image processing," in *5th Biennial Int. Conf. Nascent Technol. Eng. (ICNTE)*, Navi Mumbai, India, 2023, pp. 1–5.

[7] K. Simonyan and Z. Andrew, "Automatic imagery bank cheque data extraction based on machine learning approaches: A comprehensive survey," *Multimed. Tools Appl.*, vol. 82, no. 20, pp. 30543–30598, 2023. doi: 10.1007/s11042-023-14534-7.

[8] S. Z. M. Maung and N. Aye, "Text region localization and recognition for ID card identification using deep learning approaches," in *IEEE Conf. Comput. Appl.*, Yangon, Myanmar, 2023, pp. 411–416.

[9] C. T. Haile, "Customers identity card data detection and recognition using image processing," Ph.D. dissertation, Dept. of Computer Science, St. Mary's Univ., Addis Ababa, Ethiopia, 2023.

[10] N. Chigali, S. R. Bobba, K. Suvarna Vani, and S. Rajeswari, "OCR assisted translator," in *7th Int. Conf. Smart Struc. Syst. (ICSSS)*, Chennai, India, 2020, pp. 1–4.

[11] K. C. Shekar, M. A. Cross, and V. Vasudevan, "Optical character recognition and neural machine translation using deep learning techniques," in *Innov. Comput. Sci. Eng.: Proc. 8th ICICSE*, Singapore, 2021, pp. 277–283.

[12] V. Pomazan, I. Tvoroshenko, and V. Gorokhovatskyi, "Handwritten character recognition models based on convolutional neural networks," *Int. J. Acad. Eng. Res.*, vol. 7, no. 9, pp. 64–72, Sep. 2023.

[13] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, Apr. 2020.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.

[15] W. Liu *et al.*, "SSD: Single shot MultiBox detector," in *Computer Vision–ECCV 2016*, Las Vegas, US, 2016, pp. 21–37.

[16] K. Simonyan and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[17] C. Szegedy, *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 1–9.
[18] K. He, X. Zhang, S. Ren, and S. Jian, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.
[19] T. Mingxing and V. L. Quoc, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[20] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2010.
[21] Z. Andrew *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.
[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," arXiv preprint arXiv:1512.00567, 2015.
[23] A. Krizhevsky, I. Sutskever, and E.H. Geoffrey, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017. doi: 10.1145/3065386.
[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791.
[25] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, 2017. doi: 10.1109/TPAMI.2016.2646371.
[26] The World Bank, "Arab world population," Accessed: Dec. 24, 2023. [Online]. Available: https://data.worldbank.org.cn/
[27] S. Alghyaline, "Arabic optical character recognition: A review," *Comput. Model. Eng. Sci.*, vol. 135, no. 3, pp. 1825–1861, 2023. doi: 10.32604/cmes.2022.024555.
[28] S. Alghyaline, "A printed arabic optical character recognition system using deep learning," *J. Comput. Sci.*, vol. 18, no. 11, pp. 1038–1050, Nov. 2022. doi: 10.3844/jcssp.2022.1038.1050.
[29] Z. A. Lu, I. Bazzi, A. Kornai, J. Makhoul, P. S. Natarajan and R. Schwartz, "Robust language-independent OCR system," in *27th AIPR Workshop: Adv. Compu. Assist. Recognit.*, 1999, pp. 96–104.
[30] M. A. Alghamdi, I. S. Alkhazi, and W. J. Teahan, "Arabic OCR evaluation tool," in *7th Int. Conf. Comput. Sci. Info. Technol. (CSIT)*, Amman, Jordan, 2016, pp. 1–6.
[31] P. H. Jain, V. Kumar, J. Samuel, S. Singh, A. Mannepalli and R. Anderson, "Artificially intelligent readers: An adaptive framework for original handwritten numerical digits recognition with OCR methods," *Information*, vol. 14, no. 6, pp. 305, May 2023. doi: 10.3390/info14060305.
[32] H. E. -B. El-Sawy and M. L. Ahmed, "Arabic handwritten characters recognition using convolutional neural network," *WSEAS Trans. Comput. Res.*, vol. 5, no. 1, pp. 11–19, 2017.
[33] M. Shams, A. A. Elsonbaty, and W. Z. El Sawy, "Arabic handwritten character recognition based on convolution neural networks and support vector machine," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 8, pp. 144–149, 2020. doi: 10.14569/issn.2156-5570.
[34] J. H. Alkhateeb, "An effective deep learning approach for improving off-line arabic handwritten character recognition," *Int. J. Soft. Eng. Comput. Syst.*, vol. 6, no. 2, pp. 53–61, 2021. doi: 10.15282/ijsecs.6.2.2020.7.0076.
[35] H. M. Balaha *et al.*, "Recognizing arabic handwritten characters using deep learning and genetic algorithms," *Multimed. Tools Appl.*, vol. 80, no. 21–23, pp. 32473–32509, 2021. doi: 10.1007/s11042-021-11185-4.
[36] N. Wagaa, H. Kallel, and N. Mellouli, "Improved Arabic alphabet characters classification using convolutional neural networks (CNN)," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–16, 2022. doi: 10.1155/2022/9965426.

[37] A. Rahmatulloh, R. I. Gunawan, I. Darmawan, R. Rizal, and B. Z. Rahmat, "Optimization of hijaiyah letter handwriting recognition model based on deep learning," in *Int. Conf. Adv. Data Sci., E-learn. Info. Syst. (ICADEIS)*, Istanbul, Turkey, 2022, pp. 1–7.
[38] G. Dilar and G. Inci, "Arabic calligraphy image analysis with transfer learning," *Electrica*, vol. 24, no. 1, pp. 209, 2024.
[39] R. Dhief, R. Youssef, and A. Benazza, "An ensemble learning approach using decision fusion for the recognition of Arabic handwritten characters," in *11th Int. Conf. Pattern Recognit. Appl. Methods*, 2022, pp. 51–59.
[40] R. S. Khudeyer and N. M. Almoosawi, "Combination of machine learning algorithms and ResNet50 for Arabic handwritten classification," *Informatica*, vol. 46, no. 9, pp. 39–44, 2022. doi: 10.31449/inf.v46i9.4375.
[41] L. Berriche, A. Alqahtani, and S. RekikR, "Hybrid Arabic handwritten character segmentation using CNN and graph theory algorithm," *J. King Saud Univ.—Comput. Inf. Sci.*, vol. 36, pp. 101872, 2024. doi: 10.1016/j.jksuci.2023.101872.
[42] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprintarXiv:1603.04467, Mar. 2016.
[43] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Comput. Appl.*, vol. 33, no. 7, pp. 2249–2261, 2021. doi: 10.1007/s00521-020-05070-8.
[44] M. Torki, M. E. Hussein, A. Elsallamy, M. Fayyaz, and S. Yaser, "Window-based descriptors for Arabic handwritten alphabet recognition: A comparative study on a novel dataset," arXiv preprint arXiv:1411.3519, 2014.
[45] K. Younis and A. Khateeb, "Arabic hand-written character recognition based on deep convolutional neural networks," *Jordan. J. Comput. Inf. Technol.*, vol. 3, no. 3, pp. 186, 2017.
[46] A. Alzebdeh, M. Moneb Khaled, M. Lataifeh, and A. Elnagar, "Arabic handwritten recognition based on deep convolutional neural network," in *2nd Int. Conf. Distrib. Sens. Intell. Syst.*, 2021, pp. 271–287.
[47] H. M. Balaha, H. A. Ali, M. Saraya, and M. Badawy, "A new Arabic handwritten character recognition deep learning system," *Neural Comput. Appl.*, vol. 33, no. 11, pp. 6325–6367, 2021. doi: 10.1007/s00521-020-05397-2.
[48] F. Soumia, G. Djamel, and M. Haddad, "Handwritten Arabic character recognition: Comparison of conventional machine learning and deep learning approaches," in *Int. Conf. Reliable Info. Comm. Technol.*, Langkawi, Malaysia, 2021, pp. 1127–1138.
[49] Z. Ullah and M. Jamjoom, "An intelligent approach for Arabic handwritten letter recognition using convolutional neural network," *PeerJ Comput. Sci.*, vol. 8, pp. e995, May 2022. doi: 10.7717/peerj-cs.995.