JOIV : Int. J. Inform. Visualization, 7(2) - June 2023 330-337

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Deep Convolutional Neural Networks Transfer Learning Comparison on Arabic Handwriting Recognition System

Siti Ummi Masruroh [a,*], Muhammad Fikri Syahid [a], Firman Munthaha [a], Asep Taufik Muharram [b], Rizka Amalia Putri [a]

[a] *Faculty of Science and Technology, Syarif Hidayatullah State Islamic University Jakarta, Tangerang, 15412, Indonesia*
[b] *Department of Information Technology, Jakarta State Polytechnic, West Java, 16425, Indonesia*
Corresponding author: [*]ummi.masruroh@uinjkt.ac.id

***Abstract*—Around 27 languages and more than 420 million people worldwide use Arabic letters. That makes the Arabic language one of the most used languages. However, the Arabic language has a challenge, namely the difference in letters based on their position. Arabic handwriting recognition is important for various applications, such as education and communication. One example is during a pandemic when most education has turned digital, making recognizing students' Arabic handwriting difficult. This paper aims to create a model that can recognize Arabic handwriting by comparing several CNN architectures using transfer learning to classify Arabic, Hijja, and AHCD handwriting datasets. Transfer learning is a model that has been trained by previous datasets to other datasets and is suitable for use in models with small datasets because it can improve model accuracy even with small datasets. The datasets were split into 60%, 20%, and 20% for training, validation, and testing. Each model uses data augmentation and 50% dropout on a fully connected layer to reduce overfitting. Some of the CNN architectures used in this study to create Arabic writing recognition models are ResNet, DenseNet, VGG16, VGG19, InceptionV3, and MobileNet. The models were compiled and trained with various parameters. The best model achieved to classify AHCD and Hijja dataset is VGG16 with Adam optimizer and 0.0001 learning rate. Based on this research, it is expected to know the performance of the best model for classifying Arabic handwriting.**

***Keywords*— Arabic; recognition; CNN; transfer learning; optimizer; learning rate.**

Manuscript received 25 Jan. 2023; revised 5 Mar. 2023; accepted 27 Apr. 2023. Date of publication 30 Jun. 2023.

I. INTRODUCTION

Arabic is one of the most used languages in the world as a source of documentation. The Arabic language is challenged because of the different letters based on their position. Around 27 languages use Arabic letters; Arabic is used by more than 420 million people worldwide, making it 6th on the most-used language [1]. Arabic handwriting recognition is needed on various applications, especially in this pandemic where most education is moving to digital, making it hard to recognize Arabic handwriting from the student [2]. Therefore, this paper aims to make a model recognizing Arabic handwriting. Hopefully, this model can be meaningful to learning or teaching Arabic during this pandemic.

If it is necessary to classify the target text in some text mining methods, text in images can be processed using deep learning [3]. Based on previous papers, several methods were used to recognize Arabic handwriting, such as Convolutional Neural Network (CNN) [4], Recurrent Neural Network (RNN) [5], Long Short Term Memory (LSTM), Bidirectional Long Short Term Memory (BLSTM) [6], K-Nearest Neighbor (KNN) [7], Support Vector Machine (SVM) [2], and CNN architectures such as AlexNet, VGG-16, GoogleNet, Res50 Net, ResNexT Net, and DenseNet [1].

Several CNN architectures have been used to develop models to recognize Arabic handwriting on IFN/ENIT Arabic database [1]. The method begins by using the Hierarchical Agglomerative Clustering (HAC) algorithm to divide the database into clusters that are only loosely related to one another. Then, cluster members are ordered using the newly suggested ranking method. The ranking technique begins by computing the Pyramid Histogram of Oriented Gradients (PHoG), which is followed by the Kullback-Leibler method for determining divergence. Only the matching classes with the highest rankings are subject to the classification process. Only 11% of the entire database is used in the suggested clustering and ranking stages, which minimizes computation complexity and improves classification results. The AlexNet

architecture produced the best model in this study, which had 99% accuracy and a 0.01 learning rate. Modeling took 18.15 minutes.

Another proposed CNN-based model has also been used to develop a model to recognize Arabic handwriting on AHCD and HIJJA datasets [7]. The proposed model uses an 80% dropout and 0.001 learning rate, as well as Adam for the optimizer. The proposed model achieved 97% and 88% on the AHCD and Hijja datasets. Arabic letters and characters are recognized using Deep Convolutional Neural Network (DCNN) and SVM by comparing the input templates to the pre-stored templates using fully connected DCNN and dropout SVM.

This work also considers the correctness of the corrected categorized templates and the recognized handwritten Arabic characters while calculating the corrected classification rate (CRR). The error classification rate (ECR) is also calculated. The experimental findings of this work show that the suggested algorithm can recognize, identify, and verify the Arabic characters typed in manually. The suggested system uses a clustering method based on the K-means clustering approach to address the issue of multi-stroke in Arabic characters to identify comparable Arabic letters. Compared to the state-of-the-art, the comparative evaluation shows that the system accuracy was 95.07% CRR and 4.93% ECR. [2].

Another approach to recognizing Arabic characters is to use DCNN based on Beta-elliptic parameters and Fuzzy Elementary Perceptual Codes. This paper uses two databases, LMCA and MAYASTROUN. The proposed approach achieved 98.90% accuracy [4]. Arabic writing may now be recognized from photos of natural scenes thanks to the CNN-RNN model. This study presents a CNN-RNN model for Arabic picture text recognition with an attention mechanism. Using CNN, the model creates feature sequences from an input image. To create feature sequences in the correct order, these sequences are fed through a bidirectional RNN. The bidirectional RNN may overlook certain text segmentation preparation. Therefore, the model can choose pertinent data from the feature sequences using a bidirectional RNN with an attention mechanism to generate output. End-to-end training is carried out by an attention mechanism using a common backpropagation method. The proposed method obtains 87% accuracy on Alif and Activ databases. [5].

A cursive handwritten Arabic text recognition system has been developed using different deep-learning architectures and modeling choices. The approach starts with implementing adaptive data augmentation to promote class diversity, to prevent imbalanced data sets. This algorithm assigns a weight to each word in the database lexicon, which is calculated based on the average probability of each class in a word. The models proposed are implemented in two databases, IFN/ENIT and AHDB. The highest performance was achieved by the BLSTM model with accuracies of 98.99% and 98.10% on IFN/ENIT and AHDB databases, respectively [6].

The CNN-trained model has been used to classify the AHCD dataset, which consists of 16,800 handwritten Arabic characters that are split into 13,440 training images and 3,360 testing images. This paper uses a combination of feature extractors and a trainable classifier. The proposed model achieved a 94.9% classification accuracy rate on testing images [8].

Based on several papers that have been studied previously with various methods used. This research will create an Arabic handwriting recognition model to help recognize students' handwriting so that learning Arabic during a pandemic becomes easier. In conducting this research, the model will be created using a CNN-based architecture.

A comparison of the architectural performance of CNN in recognizing Arabic handwriting will be carried out in this study, namely VGG16, VGG19, InceptionV3, MobileNet, ResNet, and DenseNet because the method used is deep learning which will produce better performance if the dataset tends to be larger [9]. So, this study will apply the data augmentation method to increase the sample variance from the dataset. Not only that but the use of transfer learning was also applied in this study to increase the model's accuracy [10] [11]. The last method that will be used to increase the model's performance is fine-tuning. Fine-tuning is a concept of transfer learning and performs better than a manually created model [12].

So, this research does not only compare CNN architectures but also improves the performance of models created using three methods, namely data augmentation, transfer learning, and fine-tuning. The model will recognize Arabic writing with input in images of hijaiyah letters with public Hijja and AHCD datasets. It is hoped that in this study, we can find out the performance of the CNN architecture in classifying Arabic handwriting and the level of performance comparison of each CNN architecture.

This study consists of 4 sections. Section 1 contains an introduction regarding the background and research conducted. Section 2 contains the materials and methods used in the research. Section 3 contains the results and discussions. Furthermore, section 4 contains conclusions about this research.

II. Material and Method

A. CNN Architectures

CNN was first introduced by LeCun around 1980. It is one of the most used deep learning methods to process visual data [13]. The primary uses of CNN are in data analytics, natural language processing, and image and signal processing [14]. CNN had an important role in deep learning history, an example of the successful implementation of how the brain works into machine learning. CNN is also one of the models that has a good performance in commercial usage [15].

CNN is a type of feedforward neural network that can use convolutional architecture to extract features from data [16]. CNNs are based on neurons layered in the organization, making them capable of learning hierarchical representations like any other neural network model. Using weights and biases, the neurons in the layers are connected [17]. Recently, it was stated that many contributions to the CNN structure went into creating deep-learned DCNNs. By deepening the network, deep CNNs can learn additional features. However, as the network depth increases, degradation and vanishing gradient issues arise [18]. Deep learning may result in the exclusion of many crucial pieces of information when information or the gradient of input images is propagated

across numerous layers. Due to this, numerous recent publications have suggested various designs to implement the deep learning notion while attaining a short path of layers. [1]. The most popular CNN architectures are Residual Networks (ResNets), DenseNet, VGG16 architecture [1], VGG19 architecture, InceptionV3 [19], and MobileNet [20].

The number and kind of layers used in these various CNN architectures vary. These changes depend on the type of application, the volume of data, and the complexity. The input, convolution, batch normalization, pooling, dropout, and output layers are among the several types of layers. [1]. As explained in the following subsections, these architectures have been used for various purposes, notably text recognition.

1) ResNet: An architecture called ResNet has thousands of levels. Building so many layers is done to learn more complex facts accurately. The ResNets model has an advantage over other architectural models in that performance does not suffer as the design becomes more complex [21]. Degradation and other harmful effects of layering will also occur. One method for preventing degradation is ResNet. There are leftover blocks in this architecture. With ResNet, the layer will also receive input from the residual units and the direct prior layer [22]. To prevent the calculation from stacking layers without adding parameters or complexity, the identity of x is appended to the residual block's output. [1].

2) DenseNet: By substituting the dense block for the main unit in the ResNet model architecture, DenseNet architecture is created. The output of one layer in the DenseNet is broadcast to all the layers in front of it. [23]. DenseNet builds feature learning models using dense blocks as the primary building component. [24]. DenseNet connects all network levels in Dense Block to provide maximum information flow between layers [25]. L layers and L connections make up conventional CNNs. Direct connections make up L(L+1)/2 of dense CNNs. Every feature map is computed in every layer before it is used in that layer. It is regarded as a very effective remedy for the vanishing-gradient issue. Final feature maps are created by concatenating all referenced feature maps from earlier sequential layers. [1].

3) VGG16: VGG16 is one of the VGGnet models using 16 layers as its architecture. Normally VGG16 uses five convolutional blocks connected to 3 MLP classifiers. The output layer uses a sigmoid activation function when are two or fewer categories and a SoftMax activation function when there are three or more categories from the dataset [26]. On the ImageNet database, the VGG-16 network was trained. The VGG-16 network has undergone considerable training, which results in outstanding accuracy even with small image data sets [27].

4) VGG19: VGG19 is similar to VGG16 and other VGGNet variants. The difference is in 3 additional convolutional layers that help identify patterns on images [19].

5) InceptionV3: InceptionV3 architecture consists of 48 layers and the development of GoogleNet or InceptionV1. The Inception-V3 model is a deep CNN that was trained on a computer with a basic configuration [28]. This architecture comprises convolutional and fully connected (FC) layers with pooling average and max and drop out after the pooling layer. The activation function used in this architecture is batch normalization, and the loss function used is softmax [19].

6) MobileNet: MobileNet is a CNN architecture for mobile devices [29]. This architecture comprises two convolutional layers: a 3x3 depthwise convolution layer and a 1x1 pointwise convolution [30]. Counting depthwise and pointwise convolutions as separate layers, MobileNet has 28 layers [20].

B. Dataset

In this section, we will describe the datasets that are used in this paper. There are two datasets, Hijja[1] and AHCD[2]. Hijja is a free, publicly available dataset of single Arabic letters collected from Arabic-speaking school children between the ages of 7 and 12. It represents 47,434 characters written by 591 participants in different forms. Data were collected in Riyadh, Saudi Arabia, from January 2019 to April 2019 [7]. AHCD is a collection of free, public Arabic letter data. Sixty individuals wrote 16,800 characters in the dataset; their ages ranged from 19 to 40, and 90% used their right hand. [8].

C. Proposed Method

Our first step is to compare each of CNN architecture's performances with transfer learning by preparing the dataset. This paper will use the same dataset as the previous paper, those are AHCD and Hijja. This paper will also follow the same dataset split configuration which is 60% for training, 20% for validation, and 20% for testing [7].

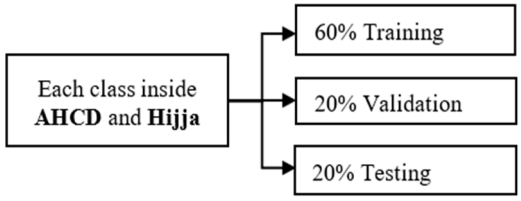

Fig. 1 Dataset Split Configuration

Each selected CNN architecture will be trained using transfer learning with pre-trained weights from *ImageNet*. Data augmentation will also be implemented on all models with the following parameters.

TABLE I
DATA AUGMENTATION CONFIGURATION FOR ALL MODELS

No	Parameter	Value
1	width_shift_range	0.2
2	height_shift_range	0.2
3	shear_range	0.2
4	zoom_range	0.2

After the dataset has been split and augmented, we will make the model from each architecture without the FC layer, in addition to our own FC layer, to classify Hijja and AHCD datasets. The FC layers are going to use 50% dropout as an attempt to reduce over-fitting. Once the model has been made, the models are going to be trained. There are two steps in training the models: training the FC layers only and for the whole layers.

[1] Hijja dataset is available at https://github.com/israksu/Hijja2

[2] AHCD dataset is available at https://www.kaggle.com/datasets/mloey1/ahcd1

The FC layers will be trained with the convolution layer frozen, meaning its weight will not be updated on training. Once the FC layers are trained, the whole layers will be trained with a small learning rate. The FC layers will be trained with 20 epochs for feature extraction, and the whole layers will be trained with 80 epochs for fine-tuning, which makes 100 epochs in total to train the whole model. There are two different parameters according to the dataset to train the model, and this whole process is transferring learning with fine-tuning.

$$Steps\ per\ epochs = \frac{Amount\ of\ dataset}{Batch\ size} \quad (1)$$

TABLE II
MODELING PARAMETER

Parameter	HIJJA		AHCD	
	Training	Validation	Training	Validation
Amount of dataset	28,460	9488	10,080	3360
Batch Size	20	16	32	32
Step per epochs	1423	593	315	105

Each model will be trained with parameters in Table 3 according to the model's dataset, and after the model has been trained, it will be re-compiled for fine-tuning with two optimizers, Adam and Stochastic Gradient Descend (SGD). Each optimizer will have three learning rates, which are 0.001, 0.0001, and 0.00001. In summary, each architecture will be re-compiled for fine-tuning with the following optimizers and learning rate configuration.

TABLE III
FINE-TUNING PARAMETERS ON EACH ARCHITECTURE

No	Optimizers	Learning Rate
1	Adam	0.001
2	Adam	0.0001
3	Adam	0.00001
4	SGD	0.001
5	SGD	0.0001
6	SGD	0.00001

Once all the models with various architectures and parameters are trained, they are going to be compared and sorted based on their *validation* and *testing accuracy*.

III. RESULT AND DISCUSSION

This paper uses two datasets Hijja, and AHCD. These datasets are divided into training, validation, and testing for 60%, 20%, and 20%, respectively. After that, the datasets will be augmented with the configuration in the previous chapter. After the dataset had been augmented, the dataset was used as an input for the model. Each model will use different selected CNN architectures, and the FC layer from each model will be trained for 20 epochs for *feature extraction* and 80 epochs for *fine-tuning*. Each model will use layer configuration in Figure 3, with 50% dropout on the FC layer and different neurons for different selected CNN architectures, see Table 5.

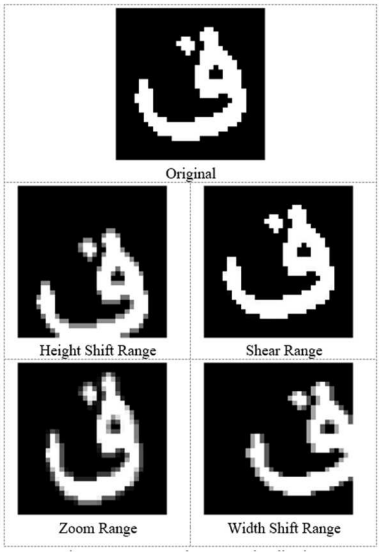

Fig. 2 Augmented Dataset Visualization

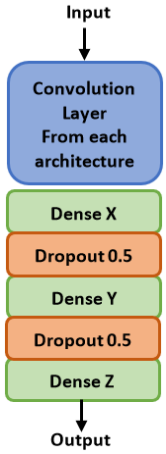

Fig. 3 Model Layer Configuration

TABLE IV
FULLY CONNECTED LAYER CONFIGURATION

Architecture	X	Y	Z	
			AHCD	Hijja
DenseNet	1024	512	28	29
InceptionV3	1024	512	28	29
MobileNet	1024	512	28	29
ResNet50	1024	512	28	29
VGG16	512	256	28	29
VGG19	512	256	28	29

After the model has been made, we train the model in 2 steps. First, we train the FC layer with the convolution layer frozen to keep the weight on the convolution layer while training the newly created FC layer. The FC layer will be trained with epochs and step per epochs from the configuration in Table 2. Once the FC layer has been trained, we re-compile the model with the parameters from Table 3 and continue the model training using the new parameter. After the training process has been done, we can compare each model's performance. The training process result with various selected CNN architectures can be seen in Table 5, Table 6, and Table 7.

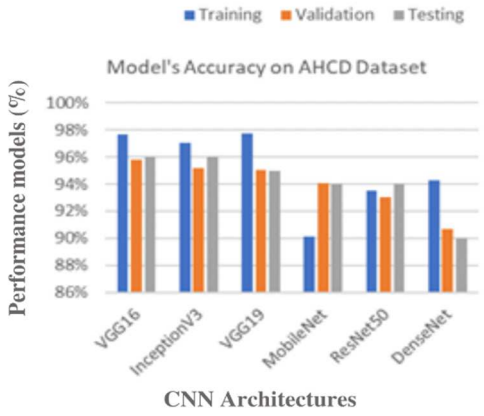

Fig. 4 Model's Accuracy on AHCD Dataset

In the AHCD dataset, VGG16 has the performance with the highest validation accuracy and testing accuracy, and VGG19 is the model with the highest training accuracy. While the model with the lowest training accuracy is MobileNet, and the model with the lowest validation accuracy and testing accuracy is DenseNet.

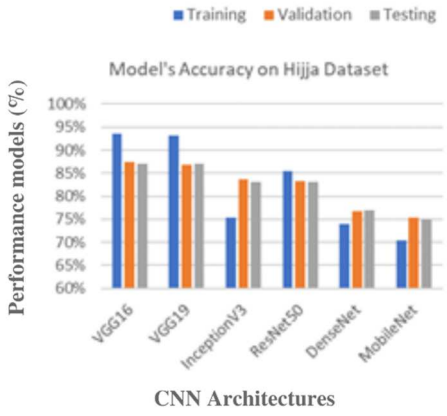

Fig. 5 Model's Accuracy on Hijja Dataset

In the Hijja dataset, VGG16 has the highest performance with training, validation, and testing accuracy. Meanwhile, MobileNet has the lowest training, validation, and testing accuracy.

TABLE V
AHCD DATASET MODEL PERFORMANCES

AHCD						
No	Architecture	Optimizer	Learning rate	Accuracy		
				Training	Validation	Testing
1	InceptionV3	ADAM	0.0001	97.02%	95.21%	96%
2	VGG16	ADAM	0.0001	97.64%	95.77%	96%
3	VGG19	ADAM	0.0001	97.73%	95.06%	95%
4	InceptionV3	ADAM	0.00001	90.97%	93.96%	94%
5	MobileNet	ADAM	0.0001	90.15%	94.05%	94%
6	ResNet50	ADAM	0.0001	93.52%	93.04%	94%
7	VGG16	ADAM	0.00001	97.13%	94.58%	94%
8	VGG19	ADAM	0.00001	97.14%	95.36%	94%
9	VGG19	SGD	0.001	96.35%	95.12%	94%
10	InceptionV3	SGD	0.001	91.35%	91.52%	92%
11	VGG16	SGD	0.0001	89.86%	91.16%	91%
12	DenseNet	ADAM	0.0001	94.26%	90.68%	90%
13	DenseNet	ADAM	0.00001	85.39%	89.13%	90%
14	VGG19	SGD	0.0001	89.08%	90.68%	90%
15	ResNet50	ADAM	0.00001	81.99%	88.51%	88%
16	DenseNet	ADAM	0.001	81.08%	87.41%	87%
17	MobileNet	ADAM	0.001	82.21%	87.65%	87%
18	MobileNet	SGD	0.001	77.63%	87.65%	87%
19	DenseNet	SGD	0.001	74.73%	85.15%	85%
20	ResNet50	SGD	0.001	70.85%	84.32%	83%
21	InceptionV3	SGD	0.0001	60.24%	82.53%	82%
22	VGG16	SGD	0.00001	78.79%	82.92%	82%
23	InceptionV3	ADAM	0.001	68.88%	79.02%	80%
24	DenseNet	SGD	0.0001	70.49%	81.01%	80%
25	ResNet50	ADAM	0.001	56.55%	73.15%	72%
26	MobileNet	ADAM	0.00001	54.70%	67.80%	67%
27	VGG16	ADAM	0.001	62.14%	66.40%	67%
28	ResNet50	SGD	0.0001	43.56%	62.26%	61%
29	DenseNet	SGD	0.00001	40.10%	53.51%	53%
30	MobileNet	SGD	0.0001	35.19%	50.36%	48%
31	InceptionV3	SGD	0.00001	19.37%	24.02%	23%
32	ResNet50	SGD	0.00001	13.88%	17.98%	17%
33	MobileNet	SGD	0.00001	11.86%	14.29%	15%
34	VGG19	ADAM	0.001	3.24%	3.57%	4%
35	VGG16	SGD	0.001	0%	0%	0%
36	VGG19	SGD	0.00001	0%	0%	0%

TABLE VI
HIJJA DATASET MODEL PERFORMANCES

HIJJA						
No	Architecture	Optimizer	Learning rate	Accuracy		
				Training	Validation	Testing
1	VGG16	ADAM	0.0001	93.59%	87.33%	87%
2	VGG19	ADAM	0.0001	93.08%	86.92%	87%
3	VGG19	ADAM	0.00001	90.05%	85.11%	85%
4	VGG19	SGD	0.001	86.75%	86.02%	85%
5	VGG16	SGD	0.001	87.60%	85.38%	85%
6	VGG16	ADAM	0.00001	90.81%	84.83%	84%
7	InceptionV3	ADAM	0.0001	75.34%	83.59%	83%
8	InceptionV3	ADAM	0.00001	78.28%	82.84%	83%
9	ResNet50	ADAM	0.0001	85.35%	83.33%	83%
10	InceptionV3	SGD	0.001	75.09%	82.06%	82%
11	ResNet50	ADAM	0.00001	77.02%	81.39%	81%
12	VGG16	SGD	0.0001	74.06%	80.99%	81%
13	VGG19	SGD	0.0001	72.60%	80.41%	80%
14	DenseNet	ADAM	0.0001	74.05%	76.77%	77%
15	DenseNet	ADAM	0.00001	67.60%	76.44%	77%
16	ResNet50	SGD	0.001	67.19%	75.41%	76%
17	MobileNet	ADAM	0.0001	70.39%	75.43%	75%
18	DenseNet	SGD	0.001	63.26%	74.28%	75%
19	InceptionV3	SGD	0.0001	59.02%	69.45%	69%
20	MobileNet	SGD	0.001	53.37%	66.91%	67%
21	VGG16	SGD	0.00001	53.71%	65.96%	66%
22	VGG19	SGD	0.00001	49.22%	62.52%	63%
23	DenseNet	SGD	0.0001	47.79%	61.42%	61%
24	MobileNet	ADAM	0.00001	44.69%	57.80%	58%
25	DenseNet	ADAM	0.001	42.83%	54.05%	54%
26	InceptionV3	ADAM	0.001	37.73%	51.62%	52%
27	DenseNet	SGD	0.00001	27.06%	40.43%	41%
28	ResNet50	SGD	0.0001	26.76%	34.49%	35%
29	InceptionV3	SGD	0.00001	24.03%	27.84%	29%
30	ResNet50	ADAM	0.001	22.93%	25.46%	27%
31	MobileNet	SGD	0.0001	11.47%	14.24%	14%
32	MobileNet	ADAM	0.001	0%	0%	0%
33	VGG16	ADAM	0.001	0%	0%	0%
34	VGG19	ADAM	0.001	0%	0%	0%
35	MobileNet	SGD	0.00001	0%	0%	0%
36	ResNet50	SGD	0.00001	0%	0%	0%

Table 5 and Table 6 show the results of the validation accuracy and testing accuracy of each model that has been trained. Based on Table 5 and Table 6, it can be seen that VGG16 with Adam optimizer and a total learning rate of 0.0001 with fine-tuning is the best architecture in classifying AHCD and hijja datasets.

TABLE VII
TOP PERFORMANCES FROM EACH CNN ARCHITECTURE

AHCD						
No	Architecture	Optimizer	Learning rate	Accuracy		
				Training	Validation	Testing
1	VGG16	ADAM	0.0001	97.64%	95.77%	96%
2	InceptionV3	ADAM	0.0001	97.02%	95.21%	96%
3	VGG19	ADAM	0.0001	97.73%	95.06%	95%
4	MobileNet	ADAM	0.0001	90.15%	94.05%	94%
5	ResNet50	ADAM	0.0001	93.52%	93.04%	94%
6	DenseNet	ADAM	0.0001	94.26%	90.68%	90%
Hijja						
No	Architecture	Optimizer	Learning rate	Accuracy		
				Training	Validation	Testing
1	VGG16	ADAM	0.0001	93.59%	87.33%	87%
2	VGG19	ADAM	0.0001	93.08%	86.92%	87%
3	InceptionV3	ADAM	0.0001	75.34%	83.59%	83%
4	ResNet50	ADAM	0.0001	85.35%	83.33%	83%
5	DenseNet	ADAM	0.0001	74.05%	76.77%	77%
6	MobileNet	ADAM	0.0001	70.39%	75.43%	75%

Table 7 shows the top models of each architecture. Based on the results, it can be seen that the model has a better overall performance on the AHCD dataset. This could be because Hijja or Arabic script made by children is more difficult to classify. This study obtained the same results as those in previous research [6], where the Adam optimizer worked better than SGD and can be seen from Table 7, where the top model for each architecture uses the Adam optimizer on fine-tuning is 0.0001. Based on the results that have been presented, the following are the differences obtained from previous relevant studies.

TABLE VIII
RESULT COMPARISON

No	Paper	Datasets	Methodology	Accuracy
1	[1]	IFN/ENIT	CNN	99%
2	[4]	LMCA, MAYASTROUN	DCNN + Beta-elliptic Parameters and Fuzzy Elementary Perceptual Codes	98.90%
3	[5]	Alif, Activ	CNN-RNN + Attention Mechanism	87%
4	[6]	IFN/ENIT, AHDB	BLSTM + Adaptive Data Augmentation	98.99% IFN/ENIT, 98.10% AHDB
5	[7]	AHCD, Hijja	CNN + Dropout + Adam Optimizer	97% AHCD, 88% Hijja
6	[8]	AHCD	CNN + Feature Extractors + Trainable Classifier	94.90%
7	This Study	Hijja, AHCD	Various CNN Algorithm + Data Augmentation + Transfer Learning + Fine-tuning	96% AHCD, 87% Hijja

IV. CONCLUSIONS

Based on the results, we can see each model's training, validation, and testing accuracy. Table 5 and 6 show the sorted model with the highest validation and testing accuracy, and Table 7 shows the top models from each architecture. Based on the results, we can see that the models have a better overall performance on the AHCD dataset. This can happen because Hijja or Arabic writing made by children is harder to classify. This paper has the same conclusion as the previous paper, which says that Adam was found to work better than SGD [7]. Table 7 shows that all the top models from each architecture use Adam optimizer instead of SGD. Another conclusion that we can conclude from observing Table 7 is that the best learning rate for CNN transfer learning architectures using Adam optimizer on fine-tuning is 0.0001. The best CNN architecture may be biased because of the same parameters for all models that might benefit only certain architecture, but with parameters from Table 3 and Table 4, we can conclude that VGG16 with Adam optimizer and 0.0001 learning rate on fine-tuning is the best architecture to classify Hijja and AHCD dataset.

REFERENCES

[1] T. M. Ghanim, M. I. Khalil, and H. M. Abbas, "Comparative Study on Deep Convolution Neural Networks DCNN-Based Offline Arabic Handwriting Recognition," *IEEE Access*, vol. 8, pp. 95465–95482, 2020, doi: 10.1109/ACCESS.2020.2994290.

[2] 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/IJACSA.2020.0110819.

[3] D. K. Muhsen, S. M. Ali, R. M. Zaki, and A. A. Ahmed, "Arguments extraction for e-health services based on text mining tools," *Period. Eng. Nat. Sci.*, vol. 9, no. 3, pp. 309–316, 2021, doi: 10.21533/pen.v9i3.2149.

[4] H. Akouaydi, S. Njah, W. Ouarda, A. Samet, M. Zaied, and A. M. Alimi, "Convolutional neural networks for online Arabic characters recognition with beta-elliptic knowledge domain," *2019 Int. Conf. Doc. Anal. Recognit. Work. ICDARW 2019*, vol. 6, pp. 41–46, 2019, doi: 10.1109/ICDARW.2019.50114.

[5] H. Butt, M. R. Raza, M. J. Ramzan, M. J. Ali, and M. Haris, "Attention-Based CNN-RNN Arabic Text Recognition from Natural Scene Images," *Forecasting*, vol. 3, no. 3, pp. 520–540, 2021, doi: 10.3390/forecast3030033.

[6] M. Eltay, A. Zidouri, and I. Ahmad, "Exploring Deep Learning Approaches to Recognize Handwritten Arabic Texts," *IEEE Access*, vol. 8, pp. 89882–89898, 2020, doi: 10.1109/ACCESS.2020.2994248.

[7] 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.

[8] M. N. Aljarrah, M. M. Zyout, and R. Duwairi, "Arabic Handwritten Characters Recognition Using Convolutional Neural Network," *2021 12th Int. Conf. Inf. Commun. Syst. ICICS 2021*, no. February, pp. 182–188, 2021, doi: 10.1109/ICICS52457.2021.9464596.

[9] D. Radovanovic and S. Dukanovic, "Image-Based Plant Disease Detection: A Comparison of Deep Learning and Classical Machine Learning Algorithms," *2020 24th Int. Conf. Inf. Technol. IT 2020*, no. February, pp. 1–4, 2020, doi: 10.1109/IT48810.2020.9070664.

[10] J. Rozaqi, A. Sunyoto, and R. Arief, "Implementasi Transfer Learning pada Algoritma Convolutional Neural Network Untuk Identifikasi Penyakit Daun Kentang Implementation of Transfer Learning in the Convolutional Neural Network Algorithm for Identification of Potato Leaf Disease," vol. 1, no. 1, 2021.

[11] R. Rismiyati and A. Luthfiarta, "VGG16 Transfer Learning Architecture for Salak Fruit Quality Classification," *Telematika*, vol. 18, no. 1, p. 37, 2021, doi: 10.31315/telematika.v18i1.4025.

[12] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput. Electron. Agric.*, vol. 161, no. February, pp. 272–279, 2019, doi: 10.1016/j.compag.2018.03.032.

[13] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "HybridSN: Exploring 3-D-2-D CNN Feature Hierarchy for Hyperspectral Image Classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 2, pp. 277–281, 2020, doi: 10.1109/LGRS.2019.2918719.

[14] D. R. Sarvamangala and R. V. Kulkarni, "Convolutional neural networks in medical image understanding: a survey," *Evol. Intell.*, vol. 15, no. 1, 2022, doi: 10.1007/s12065-020-00540-3.

[15] I. Goodfellow, Y. Bengio, and C. Aaron, *Deep Learning*. MIT Press, 2016.

[16] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, 2022, doi: 10.1109/TNNLS.2021.3084827.

[17] T. Kattenborn, J. Leitloff, F. Schiefer, and S. Hinz, "Review on Convolutional Neural Networks (CNN) in vegetation remote sensing," *ISPRS J. Photogramm. Remote Sens.*, vol. 173, no. November 2020, pp. 24–49, 2021, doi: 10.1016/j.isprsjprs.2020.12.010.

[18] Q. A. Al-Haija and A. Adebanjo, "Breast cancer diagnosis in histopathological images using ResNet-50 convolutional neural network," *IEMTRONICS 2020 - Int. IOT, Electron. Mechatronics Conf. Proc.*, vol. 50, 2020, doi: 10.1109/IEMTRONICS51293.2020.9216455.

[19] R. Sujatha, J. M. Chatterjee, N. Z. Jhanjhi, and S. N. Brohi, "Performance of deep learning vs machine learning in plant leaf disease detection," *Microprocess. Microsyst.*, vol. 80, no. November 2020, p. 103615, 2021, doi: 10.1016/j.micpro.2020.103615.

[20] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[21] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, "Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer," *Procedia Comput. Sci.*, vol. 179, no. 2019, pp. 423–431, 2021, doi: 10.1016/j.procs.2021.01.025.

[22] K. Deeba and B. Amutha, "ResNet - deep neural network architecture for leaf disease classification," *Microprocess. Microsyst.*, p. 103364, 2020, doi: 10.1016/j.micpro.2020.103364.

[23] A. M. Rafi *et al.*, "Application of DenseNet in Camera Model Identification and Post-processing Detection," pp. 19–28, 2018.

[24] Z. Zhang, Z. Tang, Y. Wang, H. Zhang, S. Yan, and M. Wang, "Compressed DenseNet for Lightweight Character Recognition," *arXiv Prepr. arXiv1912.07016*, pp. 1–11, 2019.

[25] K. Zhang, Y. Guo, X. Wang, J. Yuan, and Q. Ding, "Multiple feature reweight DenseNet for image classification," *IEEE Access*, vol. 7, pp. 9872–9880, 2019, doi: 10.1109/ACCESS.2018.2890127.

[26] P. Hridayami, I. K. G. D. Putra, and K. S. Wibawa, "Fish species recognition using VGG16 deep convolutional neural network," *J. Comput. Sci. Eng.*, vol. 13, no. 3, pp. 124–130, 2019, doi: 10.5626/JCSE.2019.13.3.124.

[27] D. Theckedath and R. R. Sedamkar, "Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks," *SN Comput. Sci.*, vol. 1, no. 2, pp. 1–7, 2020, doi: 10.1007/s42979-020-0114-9.

[28] M. Mujahid, F. Rustam, R. Álvarez, J. Luis Vidal Mazón, I. de la T. Díez, and I. Ashraf, "Pneumonia Classification from X-ray Images with Inception-V3 and Convolutional Neural Network," *Diagnostics*, vol. 12, no. 5, pp. 1–16, 2022, doi: 10.3390/diagnostics12051280.

[29] A. Michele, V. Colin, and D. D. Santika, "Mobilenet convolutional neural networks and support vector machines for palmprint recognition," *Procedia Comput. Sci.*, vol. 157, pp. 110–117, 2019, doi: 10.1016/j.procs.2019.08.147.

[30] C. Bi, J. Wang, Y. Duan, B. Fu, J. R. Kang, and Y. Shi, "MobileNet Based Apple Leaf Diseases Identification," *Mob. Networks Appl.*, 2020, doi: 10.1007/s11036-020-01640-1.