

Research Article

Human-Computer Interaction with Hand Gesture Recognition Using ResNet and MobileNet

Abeer Alnuaim ¹, **Mohammed Zakariah** ², **Wesam Atef Hatamleh**,² **Hussam Tarazi**,³ **Vikas Tripathi** ⁴, and **Enoch Tetteh Amoatey** ⁵

¹Department of Computer Science and Engineering, College of Applied Studies and Community Services King Saud University, P.O. Box 22459, Riyadh 11495, Saudi Arabia

²Department of Computer Science, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

³Department of Computer Science and Informatics, School of Engineering and Computer Science, Oakland University, 318 Meadow Brook Rd, Rochester, MI 48309, USA

⁴Department of Computer Science & Engineering, College of Graphic Era Deemed to be University, Dehradun, Uttarakhand, India

⁵School of Engineering, University for Development Studies, Tamale, Ghana

Correspondence should be addressed to Enoch Tetteh Amoatey; eamoatey@uds.edu.gh

Received 30 December 2021; Revised 30 January 2022; Accepted 7 February 2022; Published 26 March 2022

Academic Editor: Vijay Kumar

Copyright © 2022 Abeer Alnuaim et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Sign language is the native language of deaf people, which they use in their daily life, and it facilitates the communication process between deaf people. The problem faced by deaf people is targeted using sign language technique. Sign language refers to the use of the arms and hands to communicate, particularly among those who are deaf. This varies depending on the person and the location from which they come. As a result, there is no standardization about the sign language to be used; for example, American, British, Chinese, and Arab sign languages are all distinct. Here, in this study we trained a model, which will be able to classify the Arabic sign language, which consists of 32 Arabic alphabet sign classes. In images, sign language is detected through the pose of the hand. In this study, we proposed a framework, which consists of two CNN models, and each of them is individually trained on the training set. The final predictions of the two models were ensemble to achieve higher results. The dataset used in this study is released in 2019 and is called as ArSL2018. It is launched at the Prince Mohammad Bin Fahd University, Al Khobar, Saudi Arabia. The main contribution in this study is resizing the images to 64 * 64 pixels, converting from grayscale images to three-channel images, and then applying the median filter to the images, which acts as lowpass filtering in order to smooth the images and reduce noise and to make the model more robust to avoid overfitting. Then, the preprocessed image is fed into two different models, which are ResNet50 and MobileNetV2. ResNet50 and MobileNetV2 architectures were implemented together. The results we achieved on the test set for the whole data are with an accuracy of about 97% after applying many preprocessing techniques and different hyperparameters for each model, and also different data augmentation techniques.

1. Introduction

Human-computer interaction (HCI) has now become a frequent element of our lifestyles as computer technology and hardware equipment have advanced. The usage of hand signals in this HCI has piqued people's curiosity since it is a nice method of engaging with the computer [1, 2]. A hand

gesture is a means of expressing yourself via the motion of your palm and fingers [3]. Hand gesture can be used to interface with devices in HCI, eliminating any need for the additional input device. A person's gesture must be recognized by a device for effective dialogue between a human and a computer. As a result, HGR has grown in importance as a study issue and a requirement in today's environment [4],

robot control [5], virtual gaming [6], and natural user interfaces is just a few of the applications [7]. A well-known use of hand gesture recognition is the identification of human data, namely, sign language [8, 9]. Sign language is a visual language in which ideas are communicated by a series of expressive hand motions in a certain order [10]. For deaf people, it is their only means of communication. According to the World Health Organization (WHO), 5% of the world's population (about 360 million people) has moderate to severe hearing loss and can only communicate via their local sign language (WHO, 2015). Because this communication is difficult for the auditory population to grasp, there is a communication gap between the normal and speech-hearing impaired groups. As a result, computer-assisted gesture detection may be used to translate between sign languages. This would be beneficial and would act as a bridge between the two communities. Fixed movements and dynamic gestures are the two forms of sign language hand motions. Static gestures are described as the placing of hands and fingers in a fixed position with no movement in relation to time, whereas dynamic gestures are defined as hands moving in a continuous motion in relation to time. There are two methods for identifying hand gestures for sign language translation: vision-based and glove-based [11, 12]. Some of the latest works are on hand gesture recognition, which are recently published [9, 13, 14]. Electronic equipment such as data collection, accelerometers, bands, and other similar devices must be worn by participants in the contact-based method. These components detect changes in motion and transmit the data to a computer for further analysis. Although this strategy has received positive feedback in the research [15], it is costly and time-consuming for ordinary human-computer interaction interface operators. Because the signer's data are recorded using a camcorder, the vision-based technique may be more user-friendly. By analyzing data acquired through image processing algorithms, this technology decreases users' reliance on sensory equipment. This study discusses a vision-based technique for recognizing static hand movements for a sign language translation system. A novel and reliable method based on convolutional neural networks (CNNs) has been developed for this purpose. VGG-11 and VGG-16 have been improved and deployed for gesture recognition in addition to the suggested technique.

The list of contributions in this study is as follows:

- (i) The images are resized to $64 * 64$ pixels and converted from grayscale images to three-channel images.
- (ii) We applied the median filter to the images, which acts as lowpass filtering in order to smooth the images and reduce noise.
- (iii) To make the model more robust when generalizing in real-life scenarios (images) and also to avoid overfitting as possible, different augmentation techniques are applied to the images.
- (iv) Then, the preprocessed image is fed into two different models, which are ResNet50 and MobileNetV2.
- (v) ResNet50 and MobileNetV2 architectures were implemented together.

The remaining of the study is organized as follows: Section 2 gives the literature addressing the problem and the technique applied to solve the sign language recognition issue, and then, Section 3 gives the details of the dataset and different preprocessing techniques. Section 4 describes the methodology applied with proper architecture design, and Section 5 gives the results with comparative analysis followed by Section 6 with conclusions followed by the list of references.

2. Literature Review

For sign language recognition in the human-computer relationship, many approaches are present in the literature. The basic goal of these technologies is to facilitate communication by deducing the accurate interpretation of the user's gestures/signs. The following steps are included in this methodology: capture and preprocessing, gesture representation, feature extraction, and categorization. This subject looks at various movement recognition techniques in the framework of several sign languages. A summary of these strategies is provided below.

ASL Akhter (2018) described a technique for recognizing ASL alphabets. The multiple ASL alphabets are represented using PCA-based attributes, a Gabor filter, and orientation-based hash values. The collected characteristics are then categorized to use an artificial neural network (ANN). The effectiveness of their self-created database of 24 fixed motions was examined in this study. Similarly, the authors in this study [16] have created a CNN-based model for human gesture recognition. For this, the model is trained and evaluated on 31 distinct ASL alphabet and number classes. In another work [17], the authors have used a different approach of ASL alphabet identification in which both color and depth pictures of gestures were supplied into a CNN model. In this model, two convolutional layers are utilized to extract characteristics from every intake, and the information from these levels is combined and sent for categorization into a fully connected layer. In addition to RGB photographs, numerous academics have worked using depth sensors such as the Microsoft Kinect. In this study [18], the authors have demonstrated a technique for sign language recognition utilizing CNN with multiview augmentation and reasoning fusion. The depth photographs of the motions were collected using the Microsoft Kinect camera. The writers of this study proposed using augmented data for CNN model coaching. This approach obtained high classification accuracy at the expense of high computation complexity. Additional Kinect sensor-based hand gesture recognition method for ISL identification has been seen in the research [19]. In this study, a research was conducted by employing a distinct mix of feature extraction and machine learning techniques for accurate recognition of hand gestures. A total of 140 static words were used to measure performance across a variety of classes. In another work [20], the authors described a new approach for recognizing ASL

fingerspelling that makes use of a depth sensor. The writers used a principal component analysis network (PCANet) to retrieve and learn features from depth images in this document. Lastly, a linear SVM (support-vector machine) is used to classify 24 static ASL motions. Authors here [21] have suggested a method for recognizing ISL using a 5-layer CNN model. The information in this study was collected from 12 distinct signers that use the Microsoft Kinect sensor. This method's performance has only been evaluated for ISL numerals and alphabets.

For gesture recognition, several writers in journals have employed contact-based approaches. Authors here [22] have presented a wearable gadget-based approach for detecting ASL. Six inertial measuring units (IMUs) were used to create 28 ASL sentences, which were then classified using the LSTM algorithm. For a Chinese sign language translation system, Xiao et al. propose a movement detection strategy based on recurrent neural networks (RNNs). The signer's skeleton pattern is employed for bidirectional communication in this investigation. The performance of this approach is assessed using standard RGB depth images of various stationary movements. For ISL translation, [23] demonstrated a sensor-based real-time hand gesture detection system.

For segmentation, Zernike moments and SVM were utilized. Reference [24] showed a solution for static gesture recognition using Inception V3. For performance evaluation, the static image dataset of 24 English letter ASL from [25] was used. Using Inception v3, a 2-stage learning approach was used to fine tune the classification model, which achieved an efficiency of 91.35 percent. A neural network-based SLR approach was described by the researchers of [26]. A hand-detecting network based on faster R-CNN and 3D CNN for extracting features, and LSTM for encoding and decoding make up the whole identification system. Despite the fact that the dataset is limited, this method yields good identification accuracy. For SLR systems, [27] proposed a deep learning-based architecture. A CNN-based framework was used for the classification of the indications. A web camera-based database of ISL alphabets, numerals, and phrases was used to assess performance in this study. The productivity of this technique was examined using different optimizers in the CNN architecture. Contact-based, Kinect-based, and RB image-based hand motion recognitions are all evaluated in Table 1. The following conclusions may be taken from this section's literature review.

The authors in this study [28] have proposed an Arabic sign language recognition and generating Arabic speech system, they worked on a dataset, which they manually collected, the dataset has 31 classes 125 pictures for each 31 alphabet classes, and the suggested system is differently tested by combining hyperparameters to obtain the optimal outcomes with the least training time. They applied different preprocessing techniques such as the extracted images are resized to 128×128 pixels and converted to RGB and applied different augmentation techniques such as rotating images, horizontal flipping, and shearing images with 0.2 random degree range. They used a CNN model for feature extraction and classification, and the model was trained on 80% of the

dataset and tested on 20% rest of the dataset and achieved 90% accuracy on the test set.

Sagayam and Hemanth [4] proposed an Arabic sign language recognition made by a group of researchers from the Umm Al Qura University, Saudi Arabia, they worked on ArSL (2018) dataset, they applied some preprocessing techniques such as removing noise from image, converting images to gray scale, resizing them to dimensions $64 * 64$ pixels and normalizing them by dividing each pixel on 255 to convert the range of pixels from 0 to 255 to 0-1, they proposed a custom CNN model, which was trained on 80% of the data and tested on the remaining 20% (11,089 images), the accuracy they achieved on the training set was 98.06%, and the accuracy they achieved on the test set was 88.87%.

In [29], as an example, a team of academics from the Al-Azhar University in Cairo, Egypt, and the King Khalid University in Abha, Saudi Arabia, presented an Arabic sign language recognition system for alphabets using machine learning methods. They experimented on a collection of 2,800 photographs and 28 classes, with 10 persons considering the 28 distinct alphabets. Each letter received 100 photographs, for a total of 2,800 pictures. Color photographs are enhanced in order to increase picture value. The color picture is transformed to grayscale picture with 256 density stages and then resized to 640×480 pixels. Filtering techniques are also used to reduce ambient sound. They additionally conducted optimization and picture improvement algorithms, classification, and morphological straining to the color photograph in a variety of methods to produce the best example to enable us subsequently retrieve one of the best attributes and achieve the greatest efficiency. The feature extraction is performed using hand shape-based description, where each hand image was described by a vector consisting of 15 values where the values represent the key point locations, and the classification part was performed by some algorithms such as, KNN classifier, Naïve-Bayesian, and MLP. When testing, they achieved an accuracy rate of 97.548%.

3. Dataset

The dataset used is called ArSL2018 [30] launched at the Prince Mohammad Bin Fahd University, Al Khobar, Saudi Arabia, to be made available to machine learning and deep learning researchers. The dataset consists of 54,049 images for the 32 Arabic sign language sign and alphabets (32 classes), and Figure 1 shows a sample of each Arabic sign language alphabet in this dataset. Each class is named in English, and each class images vary from about 1,300 to 2,200 images, and Table 1 shows the classification of the Arabic alphabet signs, with labels and number of images.

In Figure 2 below, we can see the distribution of the classes in our dataset.

3.1. Dataset Limitations

- (1) The whole dataset was captured in a single location and on a similar background.

TABLE 1: Dataset classes and their labels and number of images.

#	Letter name in English script	Letter name in Arabic script	# of images
1	Alif	أ (ألف)	1672
2	Ba	ب (باء)	1791
3	Ta	ت (تاء)	1838
4	Tha	ث (ثاء)	1766
5	Jim	ج (جيم)	1552
6	Ha	ح (حاء)	1526
7	Kha	خ (خاء)	1607
8	Dal	د (دال)	1634
9	Dhal	ذ (ذال)	1582
10	Ra	ر (راء)	1659
11	Zay	ز (زين)	1374
12	Sin	س (سين)	1638
13	Shin	ش (شين)	1507
14	Sad	ص (داس)	1895
15	Dad	ض (ضاد)	1670
16	Ta	ط (طاء)	1816
17	Za	ظ (ظاء)	1723
18	Ayn	ع (عين)	2114
19	Ghayn	غ (غين)	1977
20	Fa	ف (فاء)	1955
21	Qaf	ق (قاف)	1705
22	Kaf	ك (كاف)	1774
23	Lam	ل (لام)	1832
24	Mim	م (ميم)	1765
25	Nun	ن (نون)	1819
26	Ha	ه (هاء)	1592
27	Waw	و (واو)	1371
28	Ya	ي (ياء)	1722
29	Taa	ة (ة)	1791
30	Ai	ا (ال)	1343
31	Laa	ال (ال)	1746
32	Yaa	ياء (ياء)	1293

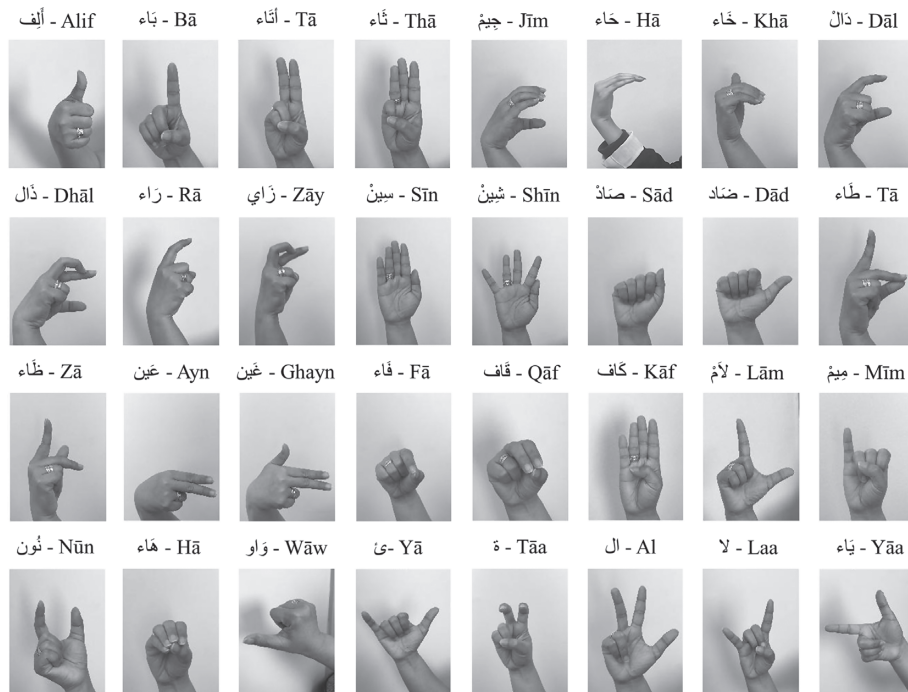


FIGURE 1: Representation of the Arabic sign language for Arabic alphabets.

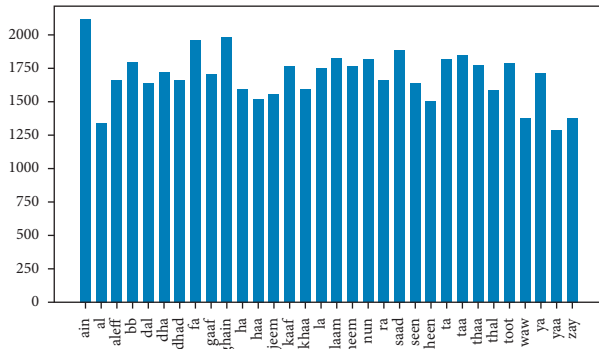


FIGURE 2: Distribution of ArSL2018 dataset classes.

- (2) There are not enough orientations, and most images in each class are taken in the same pose.
- (3) Not enough noise, lightning, and zooming variations were introduced in the classes of the dataset.
- (4) The number of contributors providing the sample images was limited.
- (5) The number of images differs from one class to another (imbalanced classes) as shown in Figure 2.

3.2. Data Preparation and Preprocessing. Data preprocessing is the first step before building a deep learning model. It is used to transform the raw data in a useful and efficient format.

3.2.1. Formatting Images. The images are resized to $64 * 64$ pixels and are converted from grayscale images to three-channel images, which are observed by stacking three grayscale channels to form the image, in order to be compatible with applying transfer learning with some models.

3.2.2. Noise Removal and Image Smoothing. We applied the median filter to the images, which acts as lowpass filtering in order to smooth the images and reduce noise. The median filter is effective at removing noise while preserving edges and features of the image.

3.3. Data Augmentation. The dataset images have many limitations as we discussed briefly in the dataset section, since all the images of the dataset nearly fall on the same distribution due to the lack of variety in the images, so we used different augmentation techniques in order to somehow address this problem, to make the model more robust when generalizing in real-life scenarios (images) and also to avoid overfitting as possible. Different augmentation techniques are applied to the images such as horizontal flip, different rotation angles, shearing, and different brightness ranges shown in Figures 3–6 respectively, random rotations are applied on images from 0 to 30° to address the problem of fixed orientations as possible, and random brightness ranges were applied in order to make the model invariant to

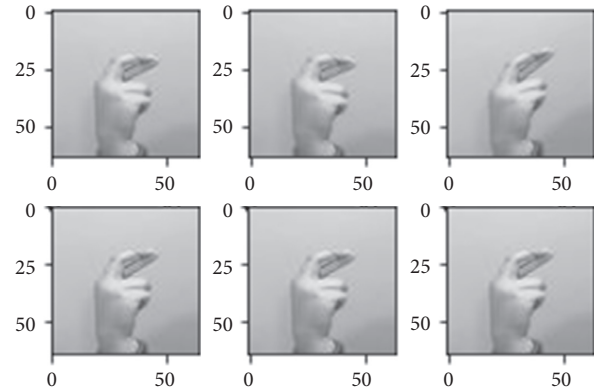


FIGURE 3: Augmentation through horizontal flipping.

different illuminations. The function used to do these operations is image flips via the `horizontal_flip` and `vertical_flip` arguments, and `rotation_range` argument `brightness_range` argument.

3.4. Splitting Dataset for Training and Testing. A ratio of 80:20 is used for dividing the dataset into training and testing set, where the constructed models were trained on around 44,000 images and tested on around 10,000 images.

4. Methodology

First, the hand is detected from the frames of the video using OpenCV, then the detected region of interest that surrounds the hand is cropped from the frame, then we apply our preprocessing techniques to the cropped hand, then the obtained cropped image is converted from RGB to gray scale, then the image is resized into dimensions $64 * 64$, then our obtained gray-scale image is converted into a three-channel image by stacking the gray-scale channel 3 times to obtain our $64 * 64 * 3$ image as we trained our models on these dimensions, then we apply the median filter to our image to reduce noise and smooth the image, and then, the preprocessed image is fed into two different models, which are ResNet50 and MobileNetV2.

4.1. ResNet50 Architecture. ResNet, short for residual networks, is a traditional neural network that helps as the basis for several computer vision applications. The essential invention with ResNet was that it empowered us to successfully train extraordinarily neural networks. Prior to ResNet, very deep neural networks were hard to train due to the issue of vanishing gradient problem.

Therefore, just tallying layers together does not work to upsurge system complexity. Deep networks are tough to train because of the well-known disappearing gradient problem. When the gradient is backpropagated to older levels, repetitive multiplication can lead the gradient to become incredibly tiny. When a consequence occurs, as the network deeply penetrates, its functionality becomes overwhelmed or even begins to degrade fast.

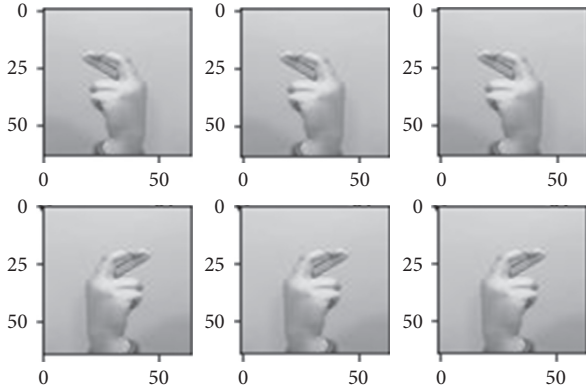


FIGURE 4: Augmentation through rotating images.

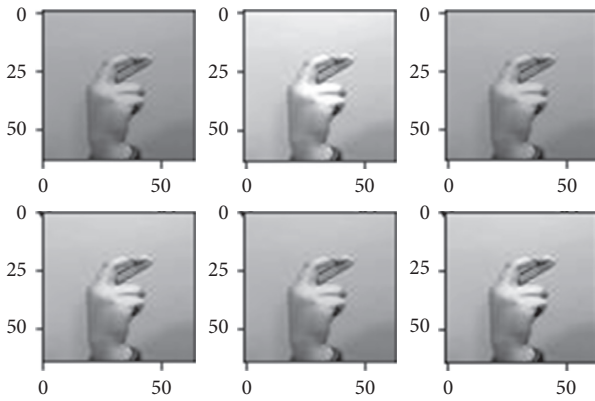


FIGURE 5: Augmentation through shearing images.

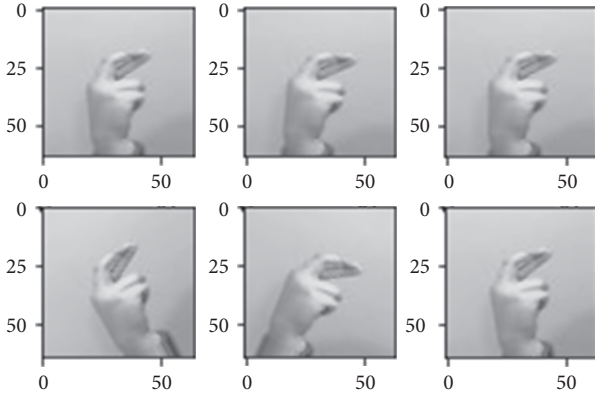


FIGURE 6: Augmentation through changing images.

When we get deeper into the model a problem occurs called vanishing gradient. So, ResNet helps in countering this issue which is its main strength. ResNet accomplishes this by applying skip connection technique. So, ResNet main strength is skip connection.

ResNet was the first to establish the notion of a skip connection. The skip link is seen in the two pictures beneath. The picture on the left, Figure 7, shows convolution layers being sequentially loaded. On the right, Figure 8, we endure to build obscurity layers as previously, but we now embrace the original data to the convolution block's output.

This identity mapping (skip connection) between layers adds the outputs from previous layers to the output of the layers ahead. This results in the ability to train much deeper networks than what was previously possible. The detailed building block of ResNet is shown in Figure 9.

Our modified ResNet50 architecture layers, hyperparameters, optimizer, etc. are shown in Table 2.

4.2. MobileNetV2 Architecture. Based on an architecture that uses depthwise separable convolutions to build lightweight deep neural networks, computation, and size trade-offs of shrinking the MobileNetV2 architecture with the width multiplier, Figure 10 shows the building block of MobileNetV2.

We all know that there are some operations that have a higher cost than other operations, for example, multiplication operation has a much higher cost than addition. Standard convolution is very costly as all its operation is multiplication as shown in Figure 11. Suppose there are N filters/kernels of size $D_k \times D_k \times M$. If a normal convolution operation is done, then the output size will be as shown in Equation (1). So, for normal convolution operation, total no of multiplications will be as shown in Equation (2). A single convolution operation requires $D_k \times D_k$ multiplications. Then, the total number of multiplications is equal to as shown in Equation (3). So,

$$\text{Total no of multiplications} = M \times D_k^2 \times D_p^2 D_p \times D_p \times N, \quad (1)$$

$$N \times D_p^2 \times D_k^2 \times M, \quad (2)$$

$$M \times D_p \times D_p \times D_k \times D_k. \quad (3)$$

$$\text{Mult sonce} = D_k^2 \times M, \quad (4)$$

$$\text{MultsperKernel} = D_G^2 \times D_k^2 \times M, \quad (5)$$

$$\text{MultsNKernels} = N \times D_G^2 \times D_k^2 \times M. \quad (6)$$

So MobileNet reduces the multiplication operation that happens in normal convolution, as MobileNet follows two main stages, which are depthwise convolution stage shown in Figure 12 and pointwise convolution stage shown in Figure 13 instead of standard convolution.

Depthwise convolution:

Pointwise convolution:

In the equations below, the detailed operations are discussed, which is used by MobileNet during the convolution stages. It is supposed there are input data of size $D_f \times D_f \times M$, where $D_f \times D_f$ can be the image size and M is the number of channels (3 for an RGB image). It is supposed that there are N filters/kernels of size $D_k \times D_k \times M$. If a normal convolution operation is performed, then, the output size will be $D_p \times D_p \times N$ as shown in equations (7)–(9).

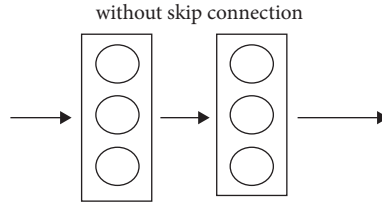


FIGURE 7: Without skip connection.

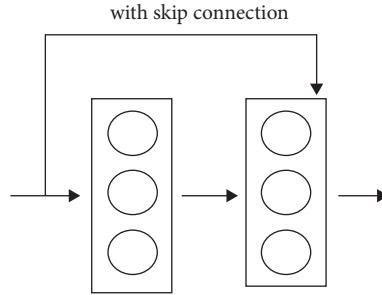


FIGURE 8: Skip connection.

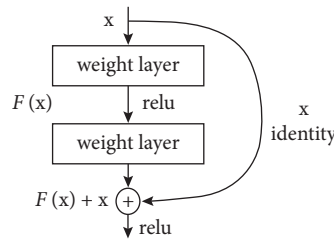


FIGURE 9: ResNet50 residual block [31].

TABLE 2: Summary of the modified ResNet50 model.

Transfer learning	Pretrained on Google ImageNet dataset
Number of layers	50 layer of ResNet50 architecture +3 sequential layers (2 fully connected layers + 1 softmax layer). Therefore, the total number of layers is 53 layers
Number of neurons in fully connected layers	1,024 neurons in the 1 fully connected layer, 512 neurons in the 2nd fully connected layer, and 32 neurons in the last layer (softmax layer)
Activation function	ReLU activation function
Optimizer	Adam optimizer with 0.0001 learning rate
Dropout	Dropout value used for the fully connected layers is 0.5
Batch size	32
Loss function	Categorical loss entropy
Number of epochs	10 epoch
Total no. of parameters	Total params: 32,495,648- trainable params: 32,450,208- nontrainable params: 45,440

In depthwise convolution step (the first step in filtering stage):

$$\text{Multsonce} = D_k^2, \quad (7)$$

$$\text{Mults1channel} = D_G^2 \times D_K^2, \quad (8)$$

$$\text{DCMults} = M \times D_G^2 \times D_K^2, \quad (9)$$

In pointwise convolution step (the second step in filtering stage):

In pointwise operation, a 1×1 convolution operation is applied on the M channels. So, the filter size for this operation will be $1 \times 1 \times M$. To say, we use N such filters, the output size becomes $D_p \times D_p \times N$, as shown in equations (10)–(13) that give the total value.

$$\text{Multsonce} = M, \quad (10)$$

$$\text{Mults1channel} = D_G \times D_G \times M, \quad (11)$$

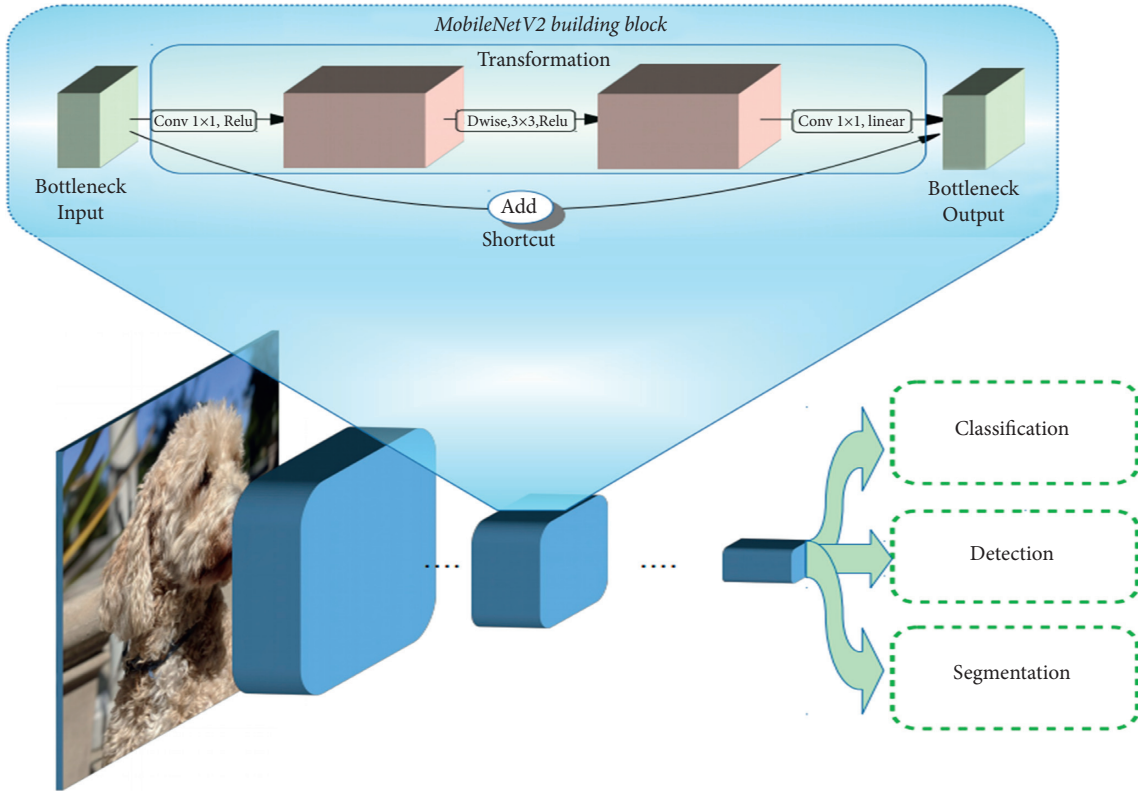


FIGURE 10: MobileNetV2 building block [32].

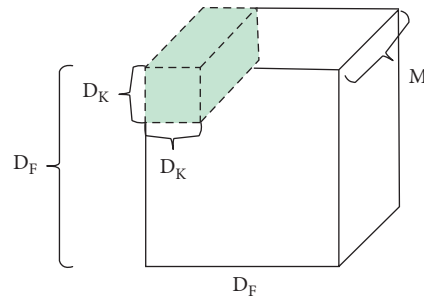


FIGURE 11: Standard convolution [33].

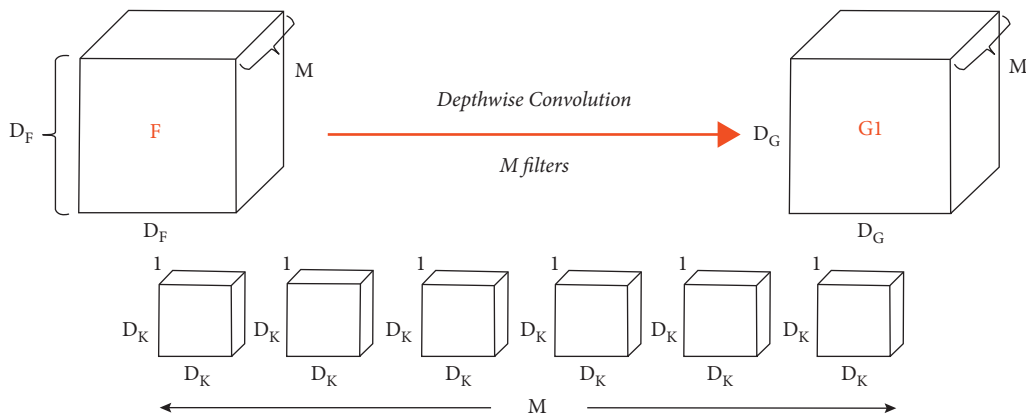


FIGURE 12: Depthwise convolution [34].

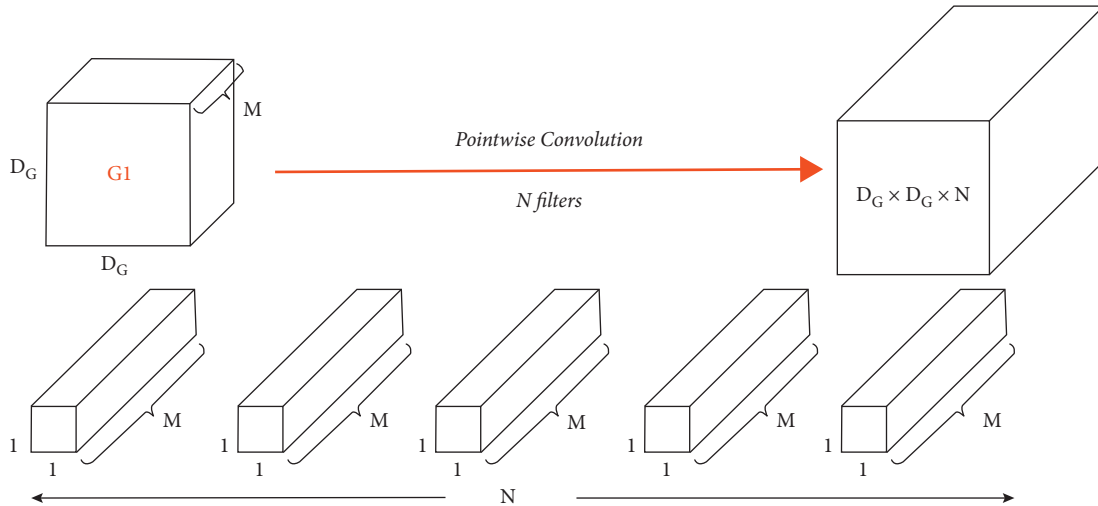


FIGURE 13: Pointwise convolution [35].

$$\text{PCMults} = N \times D_G \times D_G \times M, \quad (12)$$

$$\text{Total} = \text{DCMults} + \text{PCMults}. \quad (13)$$

Our modified MobileNetV2 architecture layers, hyper-parameters, optimizer, etc. are shown in Table 3.

4.3. Our Whole Proposed Methodology

- (1) The person hand is detected in each frame of the camera video in real time and the region of interest, in which the human hand is cropped from the video frame.
- (2) Our preprocessing techniques are applied to our cropped hand image, which are as follows: resizing the image into 64×64 , converting the image to gray scale, applying a median filter with size 3×3 for removing the salt and pepper noise generated by the camera and for image smoothing, and stacking 3 of the grayscale image to form an image of dimensions $64 \times 64 \times 3$ to be compatible with what the model was trained on.
- (3) After the hand image is cropped and pre-processed, it is fed to two models, which are as follows: ResNet50 and MobileNetV2 are pre-trained on the ImageNet dataset in order to take the advantage of transfer learning instead of beginning the model weights with random initializations. The features extracted from each model (ResNet50 and MobileNetV2) are flattened into a 1D feature vector and then entered into the fully connected layers, which is responsible for the classification part, the first layer of the fully connected layers consists of 1,024 neurons, the second layer consists of 512 neurons, and then, our last layer is the softmax layer, which distributes probabilities among the 32 Arabic sign language alphabet classes. The hyperparameters,

architecture, optimizers, etc. of the two models (ResNet50 and MobileNetV2) are discussed above in detail in Tables 2 and 3, respectively.

- (4) Each of the two models used will return its predictions. Those predictions are ensemble by averaging the predictions of the two models. At the end, the alphabet class with the maximum average prediction is returned as the classified alphabet class.

The whole methodology we talked about is briefly shown in Figure 14.

5. Results

The performance of the hand sign recognition system is evaluated using the test set where the dataset is split, which consists of 54,000 images with a ratio of 80% training set to 20% test set, which is around 44,000 samples to 10,000 samples, and our results are based on these 10,000 sample.

The two models (ResNet50 and MobileNetV2) we used achieved accuracies that range from 96% to 98%, and Figure 15 shows the accuracy of the training vs the accuracy of the validation among 10 epochs on the ResNet50 model, while in Figure 16 shows the loss of the training vs the loss of the validation among 10 epochs on ResNet50 model, as we can see there is a strong positive correlation between training accuracy and testing accuracy among the 10 epochs, so we can conclude that the model overcomes the problem of overfitting during training and achieves a test accuracy, which is almost the same as the training accuracy.

We will also see other metrics to evaluate how our model individually performs with each class through the confusion matrix. As illustrated in Figure 17, the confusion matrix is a review of the projection outcomes on a classification task. The number of correct and inaccurate guesses is summarized by counting numbers and split down by class. The confusion matrix is made up of $N \times N$ matrices, where N is the number of classes (32). Where the dark cells in the matrix correspond to low cell value and brighter cells

TABLE 3: Summary of the modified MobileNetV2 model.

Transfer learning	Pretrained on Google ImageNet dataset.
Number of layers	53 layers0 of MobileNetV2 architecture +3 sequential layers (2 fully connected layers + 1 softmax layer).Therefore, the total number of layers is 56 layers
Number of neurons in fully connected layers	1,024 neurons in the 1st fully connected layer, 512 neurons in the 2nd fully connected layer, and 32 neurons in the last layer (softmax layer)
Activation function	ReLu activation function
Optimizer	Adam optimizer with 0.0005 learning rate
Dropout	Dropout value used for the fully connected layers is 0.5
Batch size	32
Loss function	Categorical loss entropy
Number of epochs	15 epoch
Total no. of parameters	Total params: 8,043,104- trainable params: 8,008,992- nontrainable params: 34,112

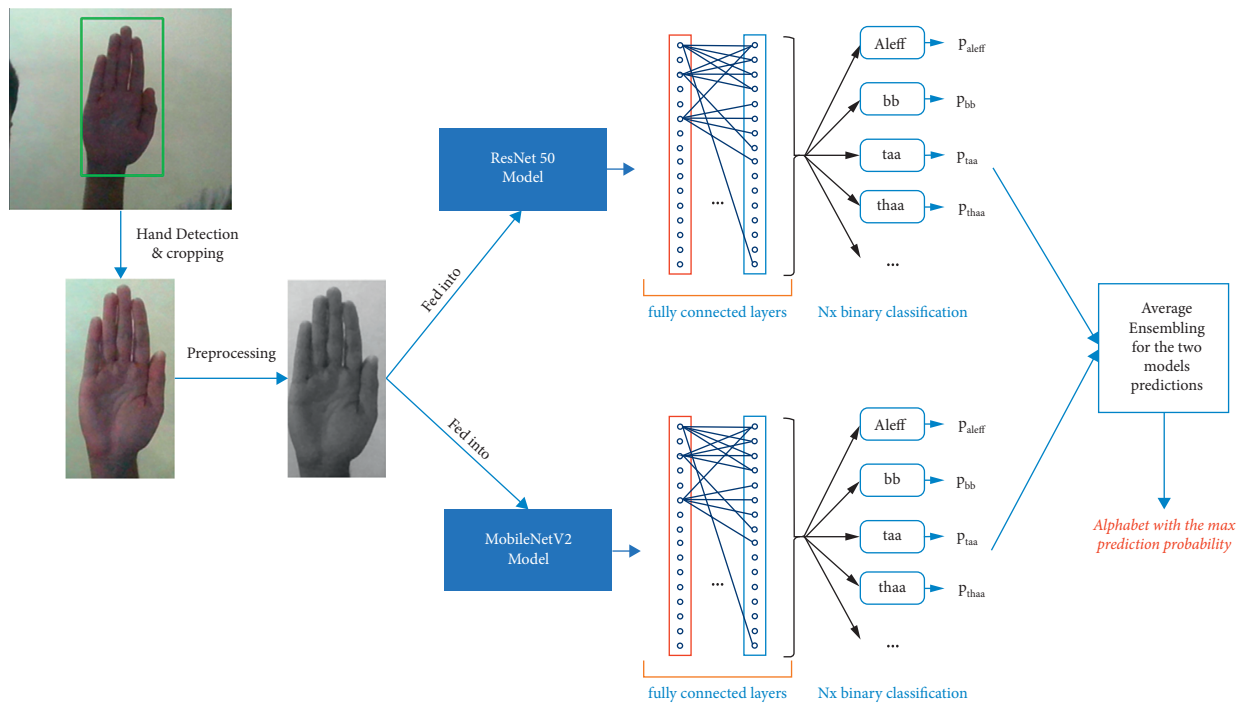


FIGURE 14: Alphabet recognition flowchart.

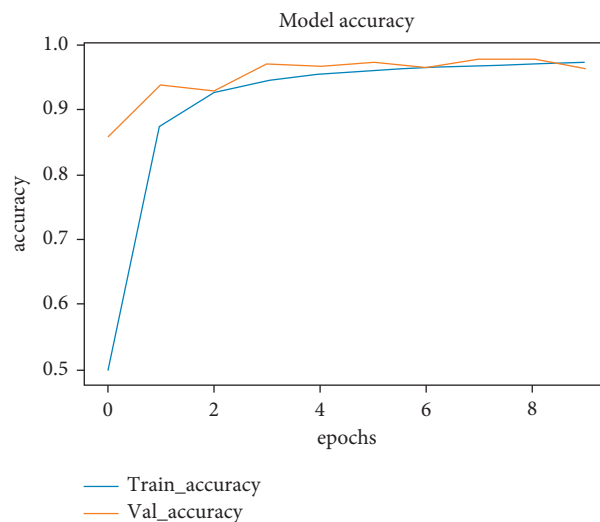


FIGURE 15: Model accuracy.

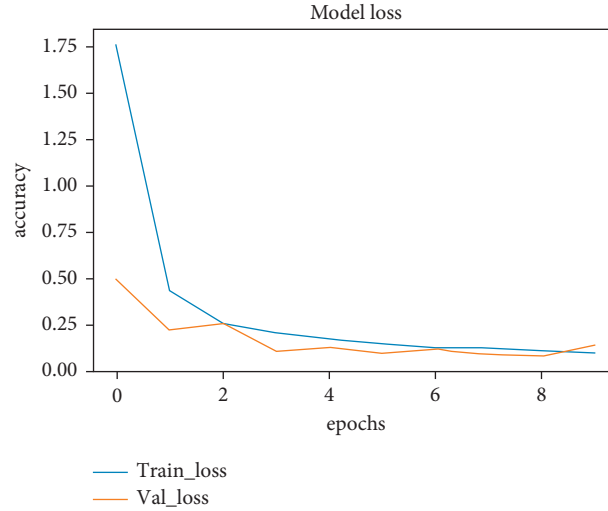


FIGURE 16: Model loss.

correspond to high cell value, we can see in Figure 17 that the diagonal has the brighter cells (highest values), which means that most of the images are correctly predicted to its real class. It shows us that our model performs well for each class on the testing set.

5.1. Evaluation Metrics. Also, we used other different evaluation metrics as the precision and recall and F1-score in order to individually evaluate our model with respect to each class from the 32 Arabic alphabet sign classes as shown in Table 4.

Precision: It is also known as the positive predictive value. Accuracy is defined as the proportion of correct predictions divided by the total number of correct class values projected. Equation (14) is used to calculate the precision.

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}. \quad (14)$$

Recall: It is sometimes referred to as vulnerability. The recall is defined as the proportion of correct predictions divided by the number of correct class values. Equation (15) is used to calculate recall.

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}. \quad (15)$$

F1-score: The *F*-score or *F*-measure is another name for the *F*-score. The *F*1-score represents the balance between precision and recall. Only when the precision and recall numbers both are good, the *F*1-score grows high. *F*1-score values array from 0 to 1, and the greater the number, the greater the classification accuracy [10].

*F*1-score is calculated by (16)

$$\text{F1 - score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (16)$$

This table summarizes the results using the metrics we described in detail above, and we can see that the combined metric, which is *F*1-score, is high for all the 32 classes, which means that our model classifies all classes as very good on the test set.

Summary of our results with different models.

The highest accuracy was received when we ensembled the prediction of our trained models, and we achieved 98.2% accuracy on the test set as shown in Table 5.

5.2. Comparative Analysis. In order to help vocal people who do not know sign language and cannot communicate with deaf people and hearing-impaired people, which mainly communicate using sign language, a lot of research has been performed for developing systems to address this problem and facilitate the communication process with deaf and hearing-impaired people. There are many research studies performed all around the world from different countries and cultures as each culture has its own sign language, which differs from other cultures. In this review section, we will focus our analysis on research studies that target Arabic sign language.

We reviewed recent several studies that are highly related to our study. Some of these studies are shown in Table 6.

6. Discussion

In this study, sign language recognition is considered for detecting the Arabic alphabet. The purpose of this study is to assist hearing-impaired people by assisting them with modern technology. The dataset used in this study is ArASL2018, and it consisted of 54,000 images of 32 Arabic sign language alphabets. Various preprocessing techniques were applied to these images like reshaping, resizing, smoothing the images, and removing the noise. Apart from preprocessing, data augmentation techniques were also applied. These techniques would help in improving the model and the accuracy. Subsequently, preprocessing is performed. The images are resized to 64 * 64 pixels and

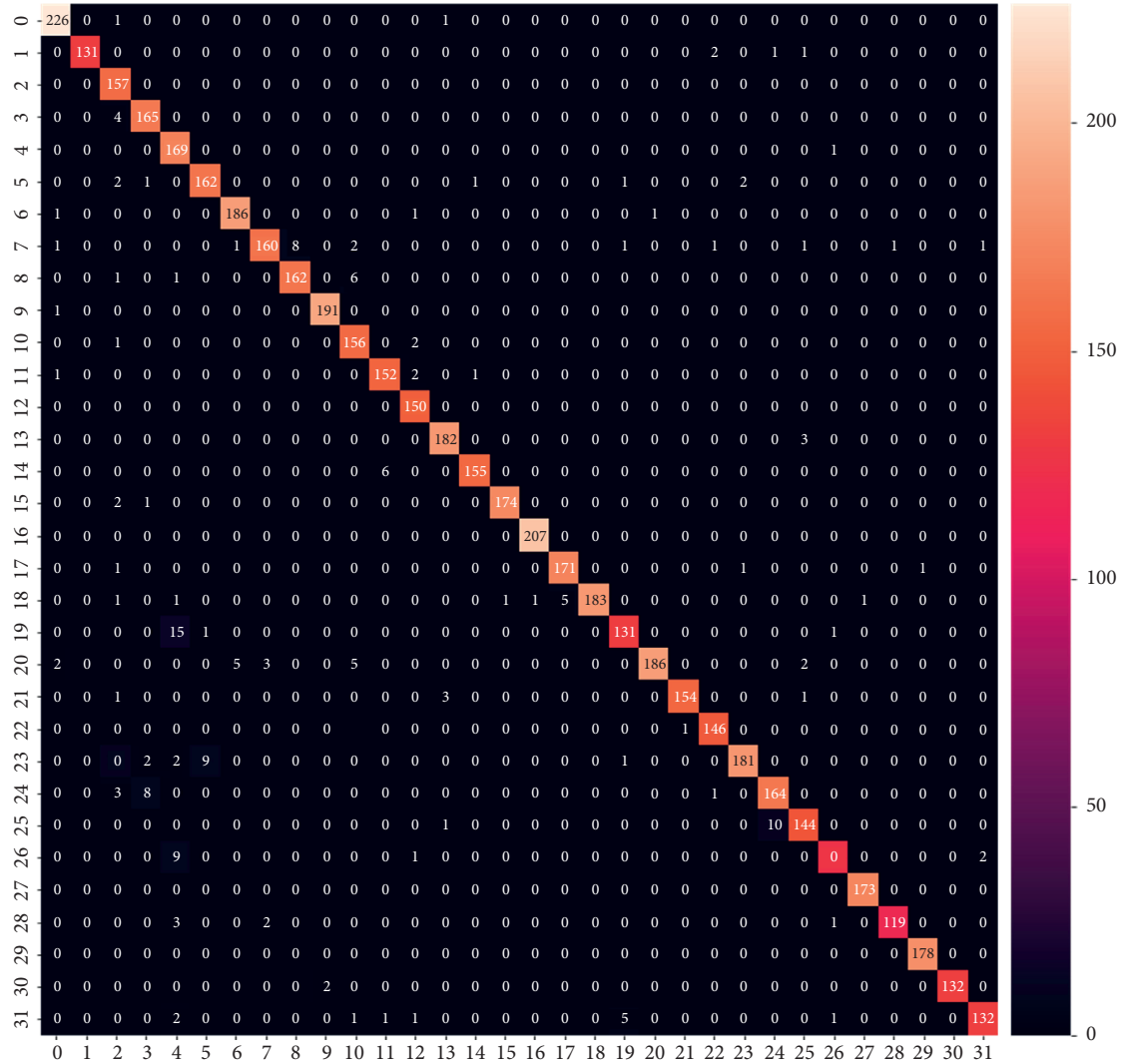


FIGURE 17: Confusion matrix.

TABLE 4: Model evaluation metrics.

Alphabets	Precision	Recall	F1-score
Ain	0.97	0.99	0.98
Al	1	0.97	0.98
Aleff	0.85	1	0.92
Bb	0.93	0.98	0.95
Dal	0.84	0.99	0.91
Dha	0.94	0.95	0.94
Dhad	0.97	0.98	0.98
Fa	0.97	0.9	0.94
Gaaf	0.95	0.95	0.95
Ghain	0.99	0.99	0.99
Ha	0.91	0.97	0.94
Haa	0.96	0.97	0.97
Jeem	0.95	1	0.97
Kaaf	0.97	0.98	0.98
Khaa	0.99	0.96	0.97
La	0.99	0.98	0.99
Laam	1	1	1
Meem	0.97	0.98	0.98

TABLE 4: Continued.

Alphabets	Precision	Recall	F1-score
Nun	1	0.95	0.97
Ra	0.94	0.89	0.91
Saad	0.99	0.92	0.95
Seen	0.99	0.97	0.98
Sheen	0.97	0.99	0.98
Ta	0.98	0.88	0.93
Taa	0.94	0.93	0.93
Thaa	0.95	0.93	0.94
Thal	0.97	0.91	0.94
Toot	0.99	1	1
Waw	0.99	0.95	0.97
Ya	0.99	1	1
Yaa	0.99	0.99	0.99
Zay	0.98	0.92	0.95

TABLE 5: Final results.

Method	Testing accuracy (%)
ResNet50	97.5
MobileNetV2	97.1
ResNet50 + MobileNetV2	98.2

TABLE 6: Comparison of related work.

Reference	Feature extraction	Classification	Accuracy (%)	Number of classes
[28]	(i) CNN Model		90	31 class
[36]	(ii) CNN Model		88.87	28 class
[29]	(i) Hand shape-based description, where each hand image was described by a vector consisting of 15 values where the values represent the key points locations	(i) KNN classifier- Naïve-Bayesian	97.548	28 class
Our research study	(i) ResNet50 and MobileNetV2		98.2	32 class

converted from grayscale images to three-channel images. We applied the median filter to the images, which acts as lowpass filtering in order to smooth the images and reduce noise, to make the model more robust when generalizing in real-life scenarios (images), and to avoid overfitting as possible. Different augmentation techniques are applied to the images. Then, the preprocessed image is fed into two different models, which are ResNet50 and MobileNetV2. ResNet50 and MobileNetV2 architectures were implemented together. We applied two different models to our experimentation, and these two models were pretrained on the Google ImageNet dataset. The first model ResNet50 achieved an accuracy on the test set of about 97.5%. The second model MobileNetV2 achieved an accuracy on the test

set of about 97%. Finally, when we ensembled the predictions of these two models, and our accuracy increased to about 98.2%.

6.1. Samples of Real-Time Detection. In Figure 18 we can see the first 9 Arabic alphabets and their classification in real time, and we process the real-time video to predict the hand gesture in each frame and classify it to one of the alphabet classes.

6.2. Limitations in Real-Time Detection. Since all the dataset was taken in the same place and under the same lighting conditions and in the static background as discussed in the



FIGURE 18: Real-time Arabic sign alphabet recognition.

dataset section, in real time the model works best on bright or white background, and static or noncomplex backgrounds.

7. Conclusions

Hand movements have been an important part of communication since the beginning of time. Sign language, a visual form of communication, is based on hand movements. An insight study of convolutional neural network (CNN) design is specifically designed for hand motion sign language identification within that research. This technique is well described and provides greater classification results with less training sets than those of other current CNN models. In this investigation, VGG-11 and VGG-16 were also designed and evaluated to test the effectiveness of this system. A currently accessible American Sign Language (ASL) database is utilized to assess competence. The competence of the proposed system, VGG-11, and VGG-16 is experimentally evaluated in comparison to state structure techniques. Several effectiveness measures, in addition to accuracy, were used to assess the efficacy of the suggested task. The findings reveal that the proposed model outperforms previous techniques since it can classify a large number of signals with a low rate of error. The technique is also tested with the new data and shown to be the rotation and resizing robust. Here in this work, we trained a model, which will be able to classify the Arabic sign language, which consists of 32 Arabic alphabet sign classes. In images, sign language is detected through the pose of the hand. In this study, we proposed a framework that consists of two CNN models, and each of them is individually trained on the training set. The final predictions of the two models were ensemble to achieve higher results. Dataset: We worked on a

dataset released in 2019 and called ArSL2018 launched at the Prince Mohammad Bin Fahd University, Al Khobar, Saudi Arabia, and we will talk about the details of this dataset in the next section. Summary of results: We achieved on the test set that we took from our whole data an accuracy of about 97% after applying many preprocessing techniques and different hyperparameters for each model, different augmentation techniques, etc.

Data Availability

The data that support the findings of this study are available in paper “Latif, Ghazanfar; Alghazo, Jaafar; Mohammad, Nazeeruddin; AlKhalaf, Roaa; AlKhalaf, Rawan (2018),” “Arabic Alphabets Sign Language Dataset (ArASL),” Mendeley Data, V1, doi: 10.17632/y7pckrw6z2.1.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors extend their appreciation to the researchers supporting project number (RSP-2021/384) of the King Saud University, Riyadh, Saudi Arabia.

References

- [1] A. Just, Y. Rodriguez, and S. Marcel, “Hand posture classification and recognition using the modified census transform,” in *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition (FG06)*, pp. 351–356, Southampton, UK, April 2006.

- [2] C.-H. Wu, W.-L. Chen, and C. H. Lin, "Depth-based hand gesture recognition," *Multimedia Tools and Applications*, vol. 75, no. 12, pp. 7065–7086, 2016.
- [3] Y. Wu and T. S. Huang, "Human hand modeling, analysis and animation in the context of HCI," in *Proceedings of the 1999 International Conference on Image Processing (Cat. 99CH36348)*, vol. 3, pp. 6–10, Kobe, Japan, October 1999.
- [4] K. M. Sagayam and D. J. Hemanth, "Hand posture and gesture recognition techniques for virtual reality applications: a survey," *Virtual Reality*, vol. 21, no. 2, pp. 91–107, 2017.
- [5] L. Tao, L. Zappella, G. D. Hager, and R. Vidal, "Surgical gesture segmentation and recognition," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 339–346, Nagoya, Japan, September 2013.
- [6] A. Kulshreshth, K. Pfeil, and J. J. La Viola, "Enhancing the gaming experience using 3D spatial user interface technologies," *IEEE Computer Graphics and Applications*, vol. 37, no. 3, pp. 16–23, 2017.
- [7] W. L. Ng, C. K. Ng, N. K. Noordin, and B. Mohd Ali, "Gesture based automating household appliances," in *Proceedings of the International Conference on Human-Computer Interaction*, pp. 285–293, Orlando, FL, USA, July 2011.
- [8] J. F. Lichtenauer, E. A. Hendriks, and M. J. Reinders, "Sign language recognition by combining statistical DTW and independent classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 2040–2046, 2008.
- [9] K. Wani and S. Ramya, "Hand gesture recognition using convex hull-based approach," in *Advances in Communication, Devices and Networking*, pp. 161–170, Springer, Singapore, 2022.
- [10] S. Sharma and S. Singh, "Vision-based sign language recognition system: a Comprehensive Review," in *Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT)*, pp. 140–144, Coimbatore, India, February 2020.
- [11] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2–3, pp. 90–126, 2006.
- [12] P. Kumar, S. S. Rautaray, and A. Agrawal, "Hand data glove: a new generation real-time mouse for Human-Computer Interaction," in *Proceedings of the 2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, pp. 750–755, Dhanbad, India, March 2012.
- [13] G. Fronteddu, S. Porcu, A. Floris, and L. Atzori, "A dynamic hand gesture recognition dataset for human-computer interfaces," *Computer Networks*, vol. 205, Article ID 108781, 2022.
- [14] M. M. Alam, M. T. Islam, and S. M. M. Rahman, "Unified learning approach for egocentric hand gesture recognition and fingertip detection," *Pattern Recognition*, vol. 121, Article ID 108200, 2022.
- [15] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 131–153, 2019.
- [16] B. Kang, S. Tripathi, and T. Q. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," in *Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 136–140, Kuala Lumpur, Malaysia, November 2015.
- [17] S. Ameen and S. Vadera, "A convolutional neural network to classify American Sign Language fingerspelling from depth and colour images," *Expert Systems*, vol. 34, no. 3, Article ID e12197, 2017.
- [18] W. Tao, M. C. Leu, and Z. Yin, "American sign language alphabet recognition using convolutional neural networks with multiview augmentation and inference fusion," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 202–213, 2018.
- [19] Z. A. Ansari and G. Harit, "Nearest neighbour classification of Indian sign language gestures using kinect camera," *Sadhana*, vol. 41, no. 2, pp. 161–182, 2016.
- [20] W. Aly, S. Aly, and S. Almotairi, "User-independent American sign language alphabet recognition based on depth image and PCANet features," *IEEE Access*, vol. 7, pp. 123138–123150, 2019.
- [21] J. Gangrade and J. Bharti, "Vision-based hand gesture recognition for Indian sign language using convolution neural network," *IETE Journal of Research*, pp. 1–10, 2020.
- [22] T.-W. Chong and B.-J. Kim, "American sign language recognition system using wearable sensors with deep learning approach," *Journal of The Korea Institute of Electronic Communication Sciences*, vol. 15, no. 2, pp. 291–298, 2020.
- [23] E. Abraham, A. Nayak, and A. Iqbal, "Real-time translation of Indian sign language using LSTM," in *Proceedings of the 2019 Global Conference for Advancement in Technology (GCAT)*, pp. 1–5, Bangalore, India, October 2019.
- [24] B. Xie, X. He, and Y. Li, "RGB-D static gesture recognition based on convolutional neural network," *Journal of Engineering*, vol. 2018, no. 16, pp. 1515–1520, 2018.
- [25] N. Pugeault and R. Bowden, "Spelling it out: real-time ASL fingerspelling recognition," in *Proceedings of the 2011 IEEE International conference on computer vision workshops (ICCV workshops)*, pp. 1114–1119, Barcelona, Spain, November 2011.
- [26] S. He, "Research of a sign language translation system based on deep learning," in *Proceedings of the 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pp. 392–396, Dublin, Ireland, October 2019.
- [27] A. Wadhawan and P. Kumar, "Deep learning-based sign language recognition system for static signs," *Neural Computing & Applications*, vol. 32, no. 12, pp. 7957–7968, 2020.
- [28] M. M. Kamruzzaman, "Arabic sign language recognition and generating Arabic speech using convolutional neural network," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 3685614, 9 pages, 2020.
- [29] G. Tharwat, A. M. Ahmed, and B. Bouallegue, "Arabic sign language recognition system for alphabets using machine learning techniques," *Journal of Electrical and Computer Engineering*, vol. 2021, Article ID 2995851, 17 pages, 2021.
- [30] G. Latif, N. Mohammad, J. Alghazo, R. AlKhalaf, and R. AlKhalaf, "ArASL: Arabic alphabets sign language dataset," *Data in Brief*, vol. 23, Article ID 103777, 2019.
- [31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.
- [32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, Salt Lake, UT, USA, June 2018.
- [33] P. K. Gadosey, Y. Li, E. A. Agyekum et al., "Sd-unet Stripping down u-net for segmentation of biomedical images on platforms with low computational budgets," *Diagnostics*, vol. 10, no. 2, p. 110, 2020.

- [34] F. Chollet, “Xception: deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, Honolulu, HI, USA, July 2017.
- [35] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, “Pointwise convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 984–993, Salt Lake, UT, USA, June 2018.
- [36] A. Althagafi, G. A. Tahani Alsubait, and T. Alqurash, “ASLR Arabic sign language recognition using convolutional neural networks,” *IJCSNS International Journal of Computer Science and Network Security*, vol. 20, no. 7, 2020.