Soft Computing (2024) 28:12991–13008
https://doi.org/10.1007/s00500-024-10305-0

NEURAL NETWORKS

Real-time mobile application for Arabic sign alphabet recognition using pre-trained CNN

Sarra Rouabhi[1] · Redouane Tlemsani[2] · Nabil Neggaz[2]

Accepted: 24 September 2024 / Published online: 19 November 2024

Abstract

The recognition of alphabet signs is pivotal for the educational, linguistic, and cognitive development of individuals. This paper delves into the application of Convolutional Neural Networks (CNNs) on the ArASL database, a compilation of Arabic alphabet sign language images. Several CNN architectures underwent testing for image classification, with the EfficientNet B7 model emerging as the most effective, achieving an outstanding test accuracy of 99.24%. Capitalizing on the success of this optimized model, we have developed a real-time application with a dual purpose: to serve as a user-friendly tool for learning Arabic sign language and to enhance communication for individuals who are deaf or hard of hearing. The application, designed for seamless integration into users' daily lives, facilitates an intuitive engagement with the Arabic sign language alphabet. Its real-time functionality empowers users to learn and communicate effortlessly, breaking down barriers and fostering inclusivity in communication. This research contributes not only to the field of computer vision and deep learning but also to the broader spectrum of inclusive education and accessibility. By harnessing cutting-edge technology, our application represents a significant step forward in leveraging artificial intelligence for the betterment of education.

Keywords Sign alphabet Arabic recognition · EfficientNet B7 · Real-time · Learning application

1 Introduction

Sign Language (SL) is an intrinsic communication system adopted by deaf and hearing-impaired individuals to facilitate interaction between themselves and with hearing individuals (Cuxac 2001; Shin et al. 2021). As a visual communication medium, SL is a language with its own distinct alphabet, lexicon, and grammar (Csizér and Kontra 2020). Its richness is expressed through the expressive use of the face, head, hands, and fingers. The need to acquire competence in SL is justified by various considerations:

(a) Facilitating communication with people who are not proficient in SL (Wanjaya et al. 2022).
(b) Encouraging interaction between deaf individuals and children born deaf (Shin et al. 2021).
(c) Highlighting the importance of SL due to the estimation that by 2050, more than 2.5 billion people worldwide will be affected by hearing loss, as reported in the first World Hearing Report of 2021 (Organization et al. 2019).

Redouane Tlemsani and Nabil Neggaz have contributed equally to this work.

✉ Sarra Rouabhi
sarra.rouabhi@univ-usto.dz

Redouane Tlemsani
redouane.tlemsani@univ-usto.dz

Nabil Neggaz
nabil.neggaz@univ-usto.dz

[1] Laboratoire de Codage et Sécurité d'Information (LACOSI), Département d'Électronique, Faculté de Génie Electrique, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf, USTO-MB, EL M'naouer, BP 1505, 31000 Oran, Algeria

[2] Dèpartementnt D'informatique, Facultè des Mathèmatique et Informatique, Universíte des Sciences et de la Technologie d'Oran Mohamed Boudiaf, USTO-MB, EL M'naouer, BP 1505, 31000 Oran, Algeria

An additional barrier to the widespread adoption of SL is the alphabetical disparity that exists from one country to another (Sutton-Spence and Woll 2004; Duarte et al. 2021).

In the realm of sign language recognition, extensive research has been conducted for various sign languages, including American English Sign Language (ESL) (Naglot and Kulkarni 2016; Rahman et al. 2019; Bird et al. 2020) Chinese Sign Language (CLS) (Jiang et al. 2020a, b), French Sign Language (FLS) (Fleurion et al. 2021; Belissen et al. 2020) among others. Pertaining specifically to the recognition of Arabic sign language (ArSL), investigations into ALS have been underway since 2005 (Mohandes and Deriche 2005).

Various techniques and algorithms have been designed to identify and recognize SLA using private databases (Mohandes and Deriche 2005; El-Bendary et al. 2010; Almasre and Al-Nuaim 2016). It is worth noting that despite these efforts, some studies in this field report recognition rates that do not exceed 70%. These results highlight the ongoing challenges in accurately recognizing SLA. In light of these limitations, researchers have increasingly turned to the application of machine learning and deep learning techniques (Alsaadi et al. 2022; Mustafa 2021) to enhance recognition capabilities.

The key contribution of our work is to accomplish a comprehensive evaluation of 12 pre-trained convolutional neural network (CNN) architectures on the publicly available ArASL2018 dataset for Arabic alphabet sign recognition. This is an important contribution as previous studies have mainly relied on private datasets, limiting the generalizability of the research. The specific objectives of this study are:

(a) Evaluate the performance of 12 pre-trained CNN models (including EfficientNet B7) on the 32-class ArASL2018 dataset, which contains the 28 standard Arabic alphabet signs as well as 4 additional signs (Latif et al. 2019).
(b) Identify the best-performing pre-trained model for recognizing Arabic alphabet signs, with the goal of integrating it into a real-time application to assist users in learning the Arabic alphabet through sign language.

Our document is organized into distinct sections. In Sect. 1, we provide an overview of the topic and outline our objectives. Section 2 offers a comprehensive summary of existing work on Arabic alphabet sign recognition, exploring various proposals. The details of our methodology, including elements such as the dataset used and pre-trained convolutional neural network models, are elucidated in Sect. 3. Section 4 is dedicated to presenting the results of our experiment, wherein we scrutinize performance metrics to evaluate the effectiveness of the proposed approach and outline the steps to achieve the real-time application of Arabic sign recognition. Section 5 highlights the limitations of real-time sign identification and the so far proposed solutions. Finally, Sect. 6 serves as the conclusion to our article, summarizing the main findings and suggesting avenues for future research.

2 Related works

This section provides a concise overview of the key advancements in Arabic alphabet sign recognition. It will discuss the databases utilized and explore the various approaches employed in the research. Our goal is to offer a comprehensive review of the field, spotlighting significant methodologies and outcomes achieved during the investigations.

Several strategies have been proposed in the existing literature to address this problem. The initial research in this area began in 2005 (Mohandes and Deriche 2005), where the researchers proposed an approach to recognize the alphabets of Arabic Sign Language (ArSL), achieving a recognition rate of 63.5%. The methodology employed in this technique is based on an in-depth examination of various descriptors. The research results highlight the superiority of the Histograms of Oriented Gradients (HOG) descriptor over others.

A system for the automatic translation of gestures into Arabic Sign Language alphabets (ArSL) was presented by El-Bendary et al. (2010). ArSL recognizes gestures using images of bare hands. The system includes five phases: preprocessing, best image detection, category detection, feature extraction, and classification. The extracted features are translation, scale, and rotation invariant. An accuracy of 91.3% using a minimum distance classifier was achieved, and 83.7% using a multi-layer perceptron.

Almasre and Al-Nuaim (2016) introduced a real-time letter recognition model for Arabic Sign Language (ArSL) using Microsoft's Kinect and Leap Motion Controller (LMC). The study aims to facilitate communication for Arabic hearing-impaired individuals by developing a system that recognizes ArSL letters in real-time. Participants were asked to gesture with the Arabic alphabet, and the model employs supervised machine learning to predict hand poses and dynamically convert these gestures into corresponding letters of the alphabet.

In 2019 (Deriche et al. 2019), an approach to recognize Arabic Sign Language was proposed. It began with acquisition for data collection, followed by a feature extraction step, including translation probability, finger length, and average tip position left. Deriche et al. classified these features using the Gauss Muscatine GMM model

and Simple Linear Discriminant Analysis (LDA). They achieved an accuracy of 92%.

In 2020, two works were published in this field. The first (Elatawy et al. 2020) aims to develop a system for recognizing the alphabet of Arabic sign language using a combination of the neutrosophic technique and the fuzzy average method. The system achieves a 91% accuracy rate in classifying the Arabic Sign Language alphabet. The second (Mustafa 2021), conducted by Mustafa et al., involves a study examining different categorization techniques used by sign language recognition systems. Many classifiers have been implemented in this context, including convolutional neural networks (CNN), recurrent neural networks (RNN), multilayer perceptron (MLP), linear discriminant analysis (LDA), Hidden Markov Models (HMM), artificial neural networks (ANN), support vector machines (SVM), and k-nearest neighbors (KNN).

Rawf et al. (2022) investigate Arabic, Persian, and Kurdish sign languages. In this study, convolutional neural networks (CNNs) and transfer learning methods are employed to categorize sign languages written in the Arabic alphabet. The goal is to establish a common standard for alphabetic sign language in these languages (Rawf et al. 2022). Additionally, a real-time Arabic Sign Language alphabet (ArSL) recognition model was presented by the authors of paper (Alsaadi et al. 2022), utilizing the AlexNet architecture. The results demonstrate that the AlexNet architecture achieved an accuracy rate of 94.81%.

Following a comprehensive analysis of the current state of research on Arabic alphabet sign recognition, it is noteworthy that a significant number of studies have primarily focused on the utilization of private databases (Mohandes and Deriche 2005; El-Bendary et al. 2010; Deriche et al. 2019). Notably, only one researcher (Alsaadi et al. 2022) has employed a complex database named ArASL2018 (Latif et al. 2019) created in 2018, encompassing 32 classes of Arabic alphabet signs.

Table 1 summarizes the related work, offering a concise overview and comparison of different approaches to Arabic alphabet sign recognition. In each row, the dataset used and the classifier employed by a different reference are detailed. The last row corresponds to our approach.

Our study makes an important contribution to the field of Arabic alphabet sign recognition by providing a comprehensive and rigorous evaluation of 12 state-of-the-art pre-trained convolutional neural network (CNN) architectures on the publicly available ArASL2018 dataset. This is a significant advancement over previous research, which has primarily relied on private or limited datasets (Mohandes and Deriche 2005; El-Bendary et al. 2010; Almasre and Al-Nuaim 2016; Deriche et al. 2019; Elatawy et al. 2020). The use of the ArASL2018 dataset, which includes 32 classes of Arabic alphabet signs, allows for a more thorough and generalizable assessment of model performance compared to studies confined to smaller private datasets.

Moreover, the systematic comparison of 12 CNN models, including the powerful EfficientNet B7 architecture, represents a more extensive evaluation than prior works that have typically focused on a narrower set of classifiers (Mustafa 2021; Rawf et al. 2022). This breadth of models assessed provides a more robust understanding of the relative merits and limitations of different deep learning approaches for Arabic alphabet sign recognition.

Importantly, the goal of identifying the best-performing pre-trained model for integration into a real-time sign language learning application is a crucial step towards developing practical, user-friendly systems to assist Arabic-speaking deaf and hard-of-hearing individuals.

3 Our methodology

This section presents an in-depth analysis of the operational framework of neural networks VGG16 and VGG19, MobileNet V1 and V2, and EfficientNet from B0 to B7, in the context of Arabic alphabet sign classification. It covers distinct categories from the ArASL2018 image data. The

Table 1 Comparison of Arabic alphabet sign recognition approaches

References	Dataset	Classifier
Mohandes and Deriche (2005)	Private Dataset	HOG
El-Bendary et al. (2010)	Private Dataset (15 letters)	MLP
Almasre and Al-Nuaim (2016)	Private Dataset	supervised machine learning
Deriche et al. (2019)	Private Dataset	GMM, LDA
Elatawy et al. (2020)	Private Dataset (300 images)	Neutrosophic technique, fuzzy average method
Mustafa (2021)	ArSL	CNN, RNN, MLP, LDA, HMM, ANN, SVM, KNN
Rawf et al. (2022)	ASSL2022	CNNs, transfer learning
Alsaadi et al. (2022)	ArASL2018	AlexNet
Our approach	ArASL2018	CNNs, EfficientNet B7

objective is to select the optimal architecture that demonstrates efficient performance in this domain. To this end, variations in image size have been applied according to the architectures considered. The final objective is the use of the optimal model for the implementation of a real-time mobile application, thus facilitating the learning of Arabic alphabet signs and communication with deaf people.

3.1 Arabic alphabets sign language dataset

The Arabic alphabet sign language (ArASL2018) dataset (Latif et al. 2019) was created at Prince Mohammed Bin Fahd University in Saudi Arabia. It comprises 54,049 images capturing hand movements corresponding to the Arabic alphabet. The dataset is categorized into 32 classes as shown in Fig. 1, with varying numbers of photos per class. This resource is readily available for deep learning and machine learning research. The images have a pixel resolution of 64 × 64 and are in grayscale, with pixel values ranging from 0 to 255.

3.2 Pre-trained model

A pre-trained convolutional neural network is a model that has undergone training on an extensive dataset to recognize and comprehend a diverse set of visual patterns and features (Qiu et al. 2020; Han et al. 2021). The advantage of utilizing pre-trained CNNs lies in their capacity to capture general visual knowledge, which can be transferred to new tasks even when the available data is limited (Zhuang et al. 2020). This proves particularly advantageous for the classification of images containing Arabic alphabet signs (Kamruzzaman 2020; Chen et al. 2020).

In our study, we evaluated various pre-trained CNN architectures, including VGG, MobileNet, and EfficientNet. These models were initially trained on large-scale image datasets such as ImageNet (Barbu et al. 2019), enabling them to comprehend a broad spectrum of objects and scenes.

3.2.1 VGG16 and 19

VGG16 and VGG19 are convolutional neural networks (CNNs) characterized by their architecture, which is based on repeated blocks of 3 × 3 convolutional layers followed by 2 × 2 max-pooling layers. Let $w_{i,j}^{l}$ (Simonyan and Zisserman 2014) represent the weight parameters of the $i-th$ filter at the $j-th$ convolutional layer. The convolution operation is denoted by *, and the rectified linear unit (ReLU) activation function by $\sigma(x) = \max(o, x)$ (Barbu et al. 2019). The forward pass-through a given layer l can be described as in Eq. 1:

$$z_i^{l+1} = \sigma\left(\sum_j W_{i,j}^{l} * z_j^{l} + b_i^{l}\right) \tag{1}$$

where z_i^{l} is the activation of the $i-th$ neuron in layer l, b_i^{l} is the bias term, and the sum is taken over all neurons j in the previous layer. Max-pooling is defined in Eq. 2:

$$p_i^{l+1} = max\left(z_{l+1}^{2i}, z_{l+1}^{2i+1}\right) \tag{2}$$

where p_i^{l+1} is the $i-th$ element of the pooled output.

The architecture includes l such blocks, and the final output is obtained through fully connected layers. The softmax function is applied to produce class probabilities (Simonyan and Zisserman 2014; Qassim et al. 2018; Theckedath and Sedamkar 2020).

The VGG16 model consists of 16 weight layers (Guan et al. 2019) (13 convolutional and 3 fully connected), as in Fig. 2, while VGG19 extends this to 19 layers, enhancing the representational capacity at the cost of increased computational complexity (Simonyan and Zisserman 2014; Lincy and Gayathri 2020).

3.2.2 MobileNet V1 and V2

MobileNet V1 is a convolutional neural network architecture designed for mobile and edge devices. It employs depthwise separable convolutions to reduce computational complexity while maintaining expressive power. Let $w_{i,j}^{l}$ represent the weight parameters of the $i-th$ filter at the $j-th$ depthwise separable convolutional layer l (Howard et al. 2017).

The depthwise separable convolution operation is defined in Eq. 3:

$$z_i^{l+1} = \sigma\left(Depthwise(w_i^{l} * z_j^{l}) + b_i^{l}\right) \tag{3}$$

where Depthwise denotes the depthwise convolution, w_i^{l} are the depthwise convolutional weights, and z_j^{l} is the activation of the $i-th$ neuron in layer l.

Pointwise convolution is then applied as in Eq. 4:

$$p_i^{l+1} = \sigma\left(\sum_j W_{i,j}^{l} * z_j^{l} + b_i^{l}\right) \tag{4}$$

where p_i^{l+1} is the $i-th$ element of the pointwise convolutional output.

MobileNet V2 builds upon MobileNet V1 by introducing inverted residuals and linear bottlenecks (Sandler et al. 2018). The inverted residual block is defined as in Eq. 5:

$$InvertedResidual(x) = ReLU6(Depthwise(w_d * ReLU6(w_s * x)) + x) \tag{5}$$

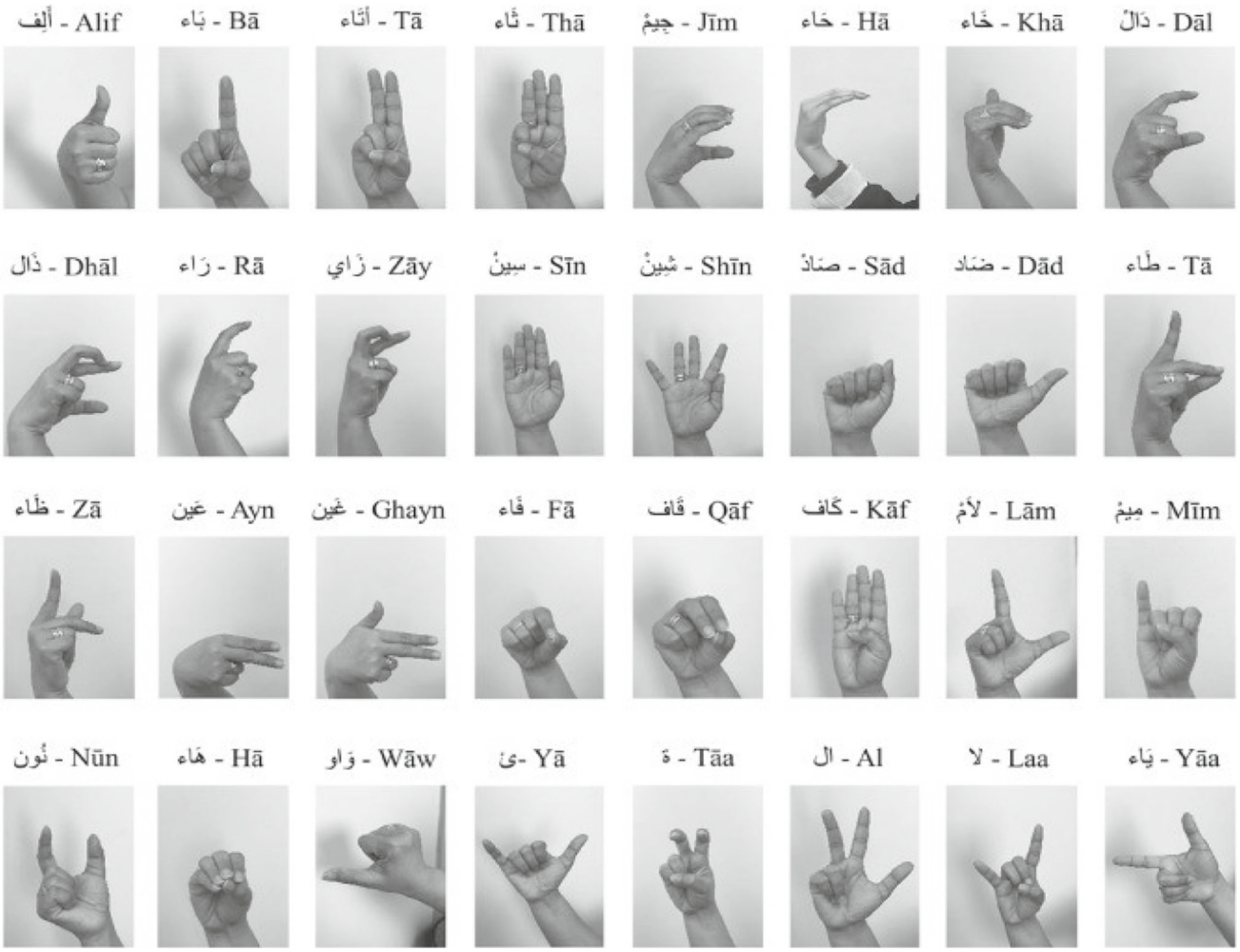
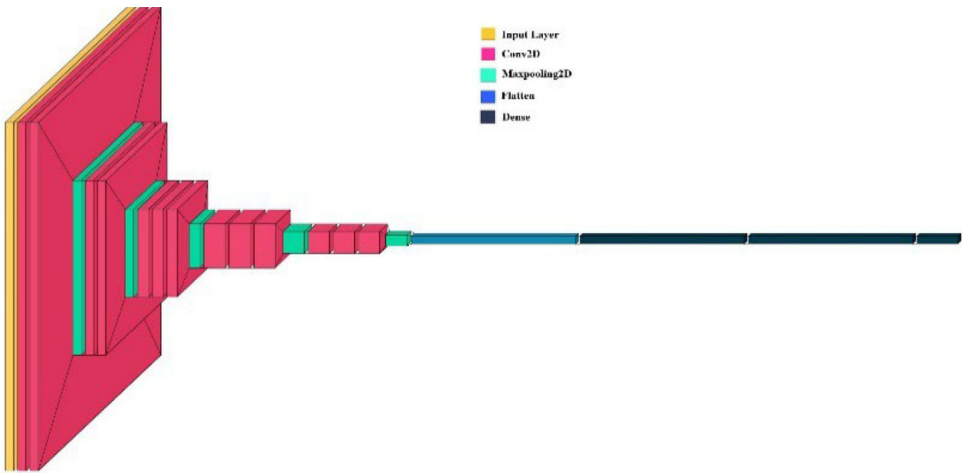


Fig. 1 Arabic alphabet sign (Latif et al. 2019)

Fig. 2 The VGG16 architecture



where w_s and w_d are the weights for the linear and depthwise convolutional layers, respectively.

The architecture includes multiple inverted residual blocks with linear bottlenecks. The final output is obtained through a combination of global average pooling and fully connected layers (Cai et al. 2020).

MobileNet V1 and MobileNet V2 as shown in Fig. 3 are designed for efficient feature extraction, making them

suitable for resource-constrained environments (Edel and Kapustin 2022; Akbar et al. 2020).

3.2.3 EfficientNet B (0 to 7)

EfficientNet EfficientNet is a family of convolutional neural network architectures that aim to achieve better accuracy and efficiency by scaling the model's depth, width, and resolution. Let $w_{i,j}^{l}$ represent the weight parameters of the $i-th$ filter at the $j-th$ layer l (Tan and Le 2019).

The architecture introduces a compound scaling method that uniformly scales the network width, depth, and resolution. The depth scaling is denoted by α, width scaling by β, and resolution scaling by γ (Tan and Le 2019; Marques et al. 2020; Kallipolitis et al. 2021).

EfficientNet employs a compound scaling formula to determine the size of the network at a given scale φ (Marques et al. 2020), as shown in Eq. 6:

$$\varphi = \alpha^{\varphi}.\beta^{\varphi}.\gamma^{\varphi} \tag{6}$$

Equation 7, shown the depth of the network:

$$d = \alpha^{\varphi} \tag{7}$$

The width of the network w is represent in Eq. 8:

$$w = \beta^{\varphi} \tag{8}$$

The resolution of the input images r is scaled as in Eq. 9:

$$r = \gamma^{\varphi} \tag{9}$$

Comparison between B0 to B7 EfficientNet comes in different scales from B0 to B7, each representing a different trade-off between model size and accuracy (Kallipolitis et al. 2021). As the scale increases from B0 to B7, the depth, width, and resolution of the network increase accordingly (Tan and Le 2019; Marques et al. 2020; Kallipolitis et al. 2021; Hoang and Jo 2021; Ali et al. 2022).

(a) B0 has the smallest model size and computational complexity (Mulim et al. 2021) as shown in Fig. 4.

(b) B7 has the largest model size and computational complexity, providing higher accuracy but requiring more resources (Ali et al. 2022).

The choice between B0 to B7 depends on the specific requirements of the application, considering factors such as available resources and desired model performance (Cenggoro 2020; Koonce and Koonce 2021).

3.3 Metrics

To determine the optimal architecture for classifying Arabic alphabet signs, we relied on metrics commonly employed by prior researchers in this field (Alsaadi et al. 2022; Rawf et al. 2022) These metrics include test accuracy as represent in Eq. 10, Macro Avg Precision as shown in Eq. 11, Macro Avg Recall as in Equation, and Macro Avg F1-score, in Eq. 12. Additionally, we computed Weighted Avg Precision, Weighted Avg Recall, and Weighted Avg F1-score, as in Eqs. 13, 14, 15, respectively, to ensure a comprehensive assessment of performance (Chicco et al. 2021; Chicco and Jurman 2020; Yi et al. 2021; Flach and Kull. 2015; Chen et al. 2018; Han and Jeong 2020). The corresponding Equations are expressed as follows:

(1) Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

(2) Macro Avg Precision:

$$Macro\,Avg\,Precision = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i} \tag{11}$$

(3) Macro Avg Recall:

$$Macro\,Avg\,Recall = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i} \tag{12}$$

(4) Macro Avg F1-score:

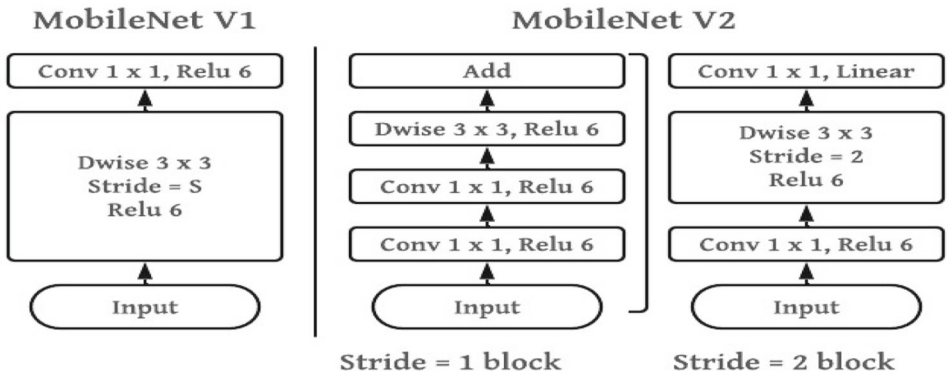


Fig. 3 MobileNet V1 and MobileNet V2 blocks (Sandler et al. 2018)

Fig. 4 EfficientNet B0 architecture (Tan and Le 2021)

$$Macro\,Avg\,F1-score = \frac{2 \times Macro\,Avg\,Precision \times Macro\,Avg\,Recall}{Macro\,Avg\,Precision + Macro\,Avg\,Recall} \quad (13)$$

(5) Weighted Avg Precision:

$$WeightedAvgPrecision = \frac{\sum_{i=1}^{N}(TP_i + FP_i)}{\sum_{i=1}^{N} TP_i} \quad (14)$$

(6) Weighted Avg Recall:

$$WeightedAvgPrecision = \frac{\sum_{i=1}^{N}(TP_i + FN_i)}{\sum_{i=1}^{N} TP_i} \quad (15)$$

(7) Weighted Avg F1-score:

$$WeightedAvgF1-score = \frac{2 \times WeightedAvgPrecision \times WeightedAvgRecall}{WeightedAvgPrecision + WeightedAvgRecall} \quad (16)$$

Here, TP denotes true positives, TN denotes true negatives, FP denotes false positives, FN denotes false negatives, and N represents the total number of classes (Yacouby and Axman 2020). These metrics provide a detailed evaluation of the model's performance in the task of Arabic alphabet sign classification.

4 Results

This section provides a detailed explanation of the operational framework of the pre-processing method employed to augment the number of images per class and elaborates on the classification of Arabic alphabet signs. It encompasses distinct classes from the (ArASL2018) image database, utilizing twelve pre-trained Convolutional Neural Networks (CNNs). This approach allows for the selection of the most effective architecture for our specific problem, thereby facilitating the choice of the most performant model for the creation of a more robust real-time application.

4.1 Pre-processing

The ArASL2018 dataset used in this study had varying numbers of images for each of the 32 Arabic alphabet classes. This imbalance in the dataset could negatively impact the learning process of the classification models. To address this issue, we implemented data augmentation techniques (Khalifa et al. 2022; Rhee et al. 2008; Ng et al. 2004) to equalize the number of training images per class.

Specifically, we divided the dataset into 80% for training, 10% for validation, and 10% for testing. We then applied the following data augmentation techniques to the training set:

- Horizontal flip: Flipping the images horizontally to create new samples.
- Rotation: Rotating the images by up to 20 degrees in either direction.
- Horizontal shift: Shifting the images horizontally by up to 20% of the image width.
- Vertical shift: Shifting the images vertically by up to 20% of the image height.
- Zoom: Zooming in on the images by up to 20%.

These augmentation techniques, as shown in Fig. 5, helped to create additional training samples and homogenize the distribution of images across the 32 alphabet classes, as illustrated in Table 2. By balancing the number of images per class, we aimed to minimize the impact of the initial data imbalance on the learning process of the classification models.

In total, 11,161 images have been generated as augmented images. Figure 6 shows some examples of the Arabic alphabet signs after augmentation.

4.2 Classification results

After pre-processing and balancing the number of images per class, the training set now contains 1700 images for each of the 32 Arabic alphabet classes. We then evaluated the performance of twelve pre-trained Convolutional

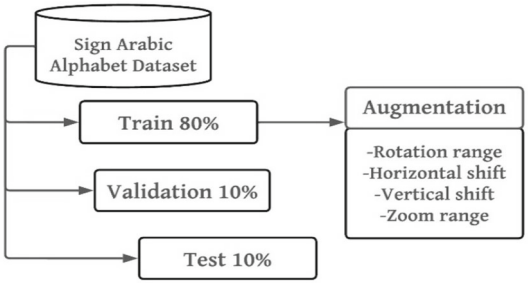


Fig. 5 Pre-processing Architecture

Table 2 Number of training images before and after augmentation for each alphabet class

Alphabet	Before augmentation	After augmentation
Alif	1516	1700
Ba	1581	1700
Ta	1413	1700
The	1465	1700
Jim	1419	1700
Haa	1412	1700
Kha	1397	1700
Dal	1074	1700
Dhal	1206	1700
Ra	1433	1700
Zay	1327	1700
Sin	1286	1700
Shin	1307	1700
Sad	1097	1700
Dad	1433	1700
Ta	1242	1700
Za	1564	1700
Ain	1378	1700
Gin	1221	1700
Fa	1378	1700
Qaf	1274	1700
Kaf	1310	1700
Lam	1364	1700
Mim	1455	1700
Noun	1453	1700
Ha	1338	1700
Waw	1336	1700
Ya	1691	1700
Taa	1470	1700
Al	1099	1700
Laa	1034	1700
Yaa	1266	1700
Total	**43,239**	**54,400**

Neural Network (CNN) architectures on this dataset, as shown in Fig. 7, to identify the most effective model for Arabic alphabet sign classification.

The hyperparameters corresponding to each pre-trained convolutional neural network (CNN) architecture are delineated in Table 3, facilitating the identification of an optimal configuration conducive to superior results attainable in real-time applications. The batch size, a crucial hyperparameter in gradient descent, prescribes the number of training samples processed before effecting modifications to the internal parameters of the model (Goodfellow et al. 2016). The Adamax optimizer has been specifically chosen for the regulation of the learning rate.

Additionally, the momentum parameter employed in the context of Batch Normalization is expressed by the Equation:

$$x = keras.layers.BatchNormalization(axis = -1, momentum = 0.99, epsilon = 0.001)(x) \quad (17)$$

where x represents the input tensor subjected to Batch Normalization. The momentum is set to 0.99, and the epsilon value is 0.001, which are common settings used to stabilize the Batch Normalization process.

Table 4 presents a comprehensive comparison of the twelve pre-trained models, highlighting their performance on the ArASL2018 dataset. Key metrics such as training accuracy, validation accuracy, best performance epoch, and execution time are evaluated. The results demonstrate that the EfficientNet B7 model stands out as the best-performing architecture, achieving an exceptional training accuracy of 99.89% and a validation accuracy of 99.45%. Notably, EfficientNet B7 reached this high level of performance as early as the 17th training epoch, indicating its efficiency and suitability for real-time applications.

Although the execution time of 147 min for EfficientNet B7 is slightly longer than some other models, this factor can be overlooked given the superior classification performance. These findings underscore the importance of carefully selecting the most appropriate pre-trained model for the specific task of Arabic alphabet sign recognition, with EfficientNet B7 emerging as the optimal choice.

The learning curves depicted in Fig. 8 further support the effectiveness of the evaluated models. The twelve subfigures show the evolution of training accuracy, validation accuracy, training loss, and validation loss over the training epochs. An important observation is that the models exhibit convergence in learning the discriminative features of Arabic alphabet signs from the 10th epoch onwards, indicating the stability of the learning process.

Fig. 6 Examples of images after the augmentation

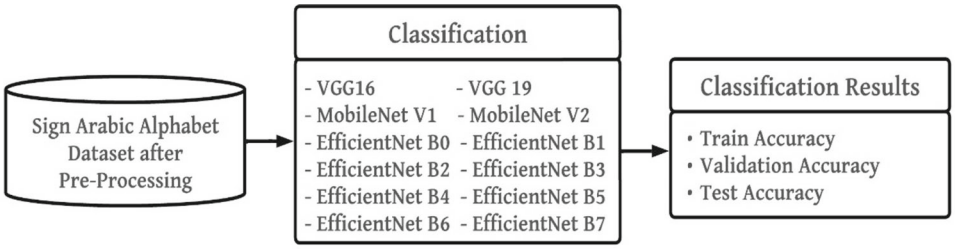


Fig. 7 The framework to compare pre-trained CNN

Table 3 Hyperparameters for experiments

Hyperparameters	VGG	MobileNet	EfficientNet
Number of epochs	20	20	20
Batch size	30	30	30
Optimizer	Adamax	Adamax	Adamax
Initial learning rate	0.001	0.001	0.001
Final learning rate	0.0005	0.0005	0.0005
Momentum	0.99	0.99	0.99
Input image shape	64 × 64	64 × 64	64 × 64
Activation function	Softmax	Softmax	Softmax

4.3 Select the best Pre-trained model

The comparative analysis of model performance in the context of Arabic alphabet sign classification, using the ArASL2018 dataset, was carried out based on key evaluation metrics such as precision, recall, F1-score, and test accuracy. These metrics are crucial in the field of classification, as they enable us to assess a model's ability to effectively discriminate between different classes.

Table 5 summarizes the results obtained from the experiments, highlighting the performance of different models, including VGG16, VGG19, MobileNet V1, MobileNet V2, and the various EfficientNet variants (B0 to B7). The metrics examined include test accuracy, Macro Average Precision, Macro Average Recall, and Macro Average F1-Score.

The results show that the EfficientNet B7 stands out as the model offering the best overall performance among all the evaluated architectures. Specifically, the EfficientNet B7 achieved the highest test accuracy rate at 99.24%, outperforming the other models. Additionally, the Macro Average Precision (99.27%), Macro Average Recall (99.25%), and Macro Average F1-Score (99.26%) for the EfficientNet B7 are also the highest across all the examined models.

These results demonstrate the superiority of the EfficientNet B7 model in the specific context of classifying Arabic alphabet signs from the ArASL2018 dataset. The model's exceptional performance across multiple evaluation metrics, including the highest test accuracy and Macro Average scores, highlights its suitability and effectiveness for this task.

The implications of these findings are significant, as the choice of the right pre-trained model is crucial for developing robust and efficient real-time applications for Arabic alphabet sign recognition. The EfficientNet B7 architecture, with its exceptional classification capabilities, can serve as a strong foundation for future research and the deployment of practical solutions in this domain.

Table 6 presents the top 10 classification errors in a model designed to recognize 32 classes in the ArASL2018 dataset. These classification errors can be attributed mainly to the incorrect formation of hand gestures rather than the model's performance. Even though these errors exist, they are relatively few (the highest being 6 errors for a specific misclassification), indicating that the model is well-trained

Table 4 Model performance metrics

Model	Train Accuracy (%)	Validation Accuracy (%)	Best Epoch (%)	Execution Time (%)
VGG16	99.71	99.20	17	41 mn
VGG19	99.66	99.20	20	30 mn
MobileNet V1	99.73	99.27	14	25 mn
MobileNet V2	99.81	99.20	18	22 mn
EfficientNet B0	99.74	99.28	18	45 mn
EfficientNet B1	99.70	99.35	20	44 mn
EfficientNet B2	99.75	99.31	17	50 mn
EfficientNet B3	99.78	99.42	19	60 mn
EfficientNet B4	99.80	99.31	18	64 mn
EfficientNet B5	99.80	99.37	20	84 mn
EfficientNet B6	99.81	99.27	17	120 mn
EfficientNet B7	99.89	99.45	17	147 mn

overall. The EfficientNet B7 performs accurately across the majority of the 32 classes.

Table 7 provides a detailed analysis of the classification performance. Each row of the table represents an Arabic alphabet class, with the following metrics evaluated for each class: precision, recall, F1 score, and support, which represents the number of occurrences of that class in the dataset.

Looking at these results, it is clear that EfficientNet B7 performs exceptionally well across all classes, with high values for precision, recall, and F1 score. In particular, some classes such as 'ain', 'khaa', 'waw', 'ya', and 'yaa' show perfect results with 100% precision, recall, and F1 score.

Table 8 represents the overall test accuracy of the model as 99.24%, underlining its excellent ability to classify Arabic alphabet signs correctly. The Weighted Avg (weighted average by the number of occurrences of each class) values further reinforce the overall excellence of the model, with high scores of 99.27%, 99.25%, and 99.26% respectively. This high level of performance underlines the relevance of the EfficientNet B7 and its usefulness in real-life image recognition applications.

Figure 9 represents the misclassification errors observed in the test set for each specific class. In particular, the 'dha' class stands out with seven misclassified images, closely followed by the 'kaaf' class with four misclassified images. The 'thal', 'tha', and 'gaaf' classes each have three misclassified images. It is essential to stress that these misclassifications can be attributed to a variety of factors, such as annotation errors in the database or variations in the way the signs were formed by different volunteers.

It should be noted that a significant source of error comes from the presence of incorrectly categorized images in the database. For example, images representing the alphabet sign 'tha' were found in the class 'gaaf'. In addition, errors can also arise from the diversity of sign formation styles used by volunteers, introducing significant variations into the data.

4.4 Application for Arabic alphabet signs

In this part, we delve into the intricate details of the development and deployment process,

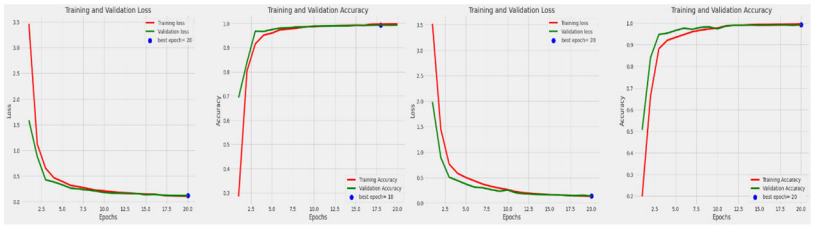
highlighting key steps of our real time application.

4.4.1 Model conversion and adaptation

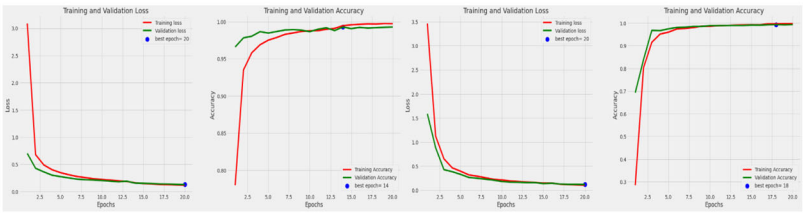
The foundation of our application lies in the conversion of the EfficientNet B7 model from TensorFlow to TensorFlow Lite. This crucial step, depicted in Fig. 10, serves the purpose of optimizing the model for mobile deployment on the Android platform (Fadlilah and Handaga 2021; Ahmed and Bons 2020; Pratama and Pratama 2021). The model, now saved as a '.h5 file,' undergoes this transformation to ensure its efficiency and seamless integration within the Android environment.

The conversion to TensorFlow Lite is essential for achieving optimal performance, enabling a smooth user experience and real-time functionality on mobile devices. This adaptation aligns with the stringent requirements for deploying applications on the Android platform, ensuring that our Arabic alphabet sign recognition model is both effective and user-friendly. Additional steps include:

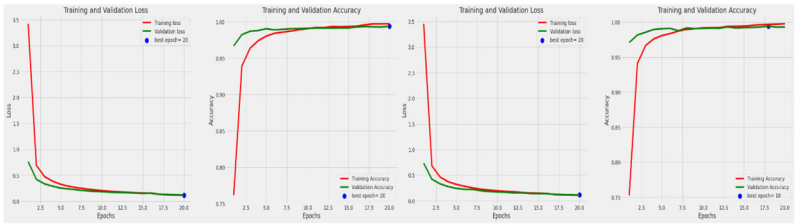
- Quantization: Reducing the model size and improving latency by converting the weights from floating-point to a more efficient data type, such as 8-bit integers.



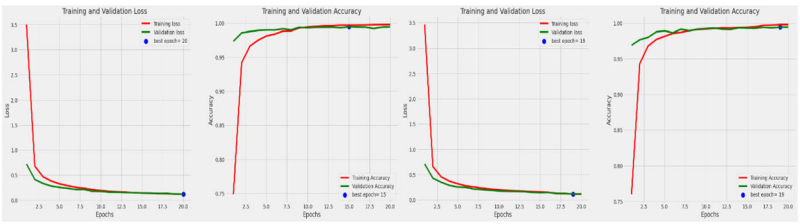
a) VGG16 b) VGG19



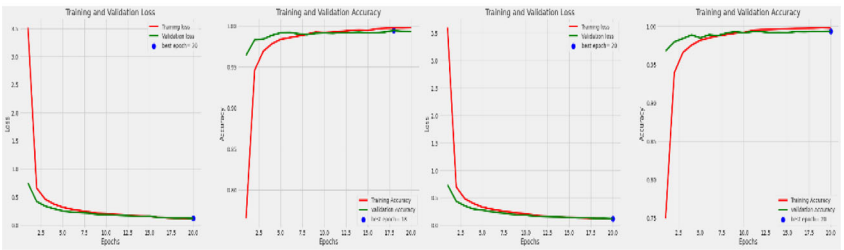
c) MobileNet V1 d) MobileNet V2



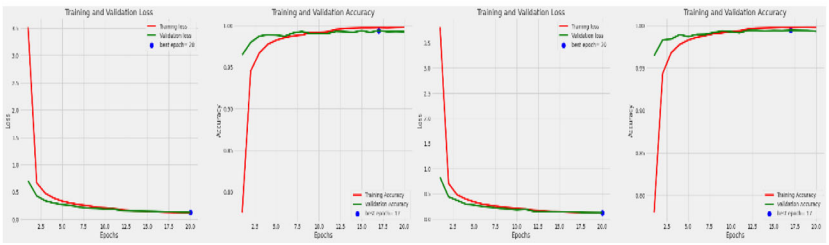
e) EfficientNet B0 f) EfficientNet B1



g) EfficientNet B2 h) EfficientNet B3



i) EfficientNet B4 j) EfficientNet B5



k) EfficientNet B6 l) EfficientNet B7

Fig. 8 Train and validation accuracy, train and validation loss for each model

Table 5 Performance comparison of various pre-trained Models in Arabic alphabet sign classification

Model	Test accuracy (%)	Macro AVG Precision (%)	Macro AVG Recall (%)	Macro AVG F1-Score (%)
VGG16	98.24	98.12	98.12	98.12
VGG19	98.25	98.26	98.24	98.25
MobileNet V1	98.98	99.02	98.98	99.00
MobileNet V2	99.06	99.09	99.06	99.07
EfficientNet B0	99.07	99.10	99.08	99.09
EfficientNet B1	99.15	99.19	99.16	99.17
EfficientNet B2	99.20	99.24	99.21	99.22
EfficientNet B3	99.20	99.24	99.21	99.22
EfficientNet B4	99.17	99.20	99.17	99.18
EfficientNet B5	99.17	99.20	99.18	99.18
EfficientNet B6	99.20	99.24	99.21	99.22
EfficientNet B7	**99.24**	**99.27**	**99.25**	**99.26**

Table 6 The top 10 classification errors

Real classes	Predicted classes	Number
dha	sheen	6
kaaf	ta	4
dhad	ra	2
al	ghain	1
dal	ra	1
dha	meem	1
fa	thal	1
gaaf	fa	1
ghain	seen	1
jeem	dal	1

- Model Optimization: Utilizing TensorFlow Lite's optimization tools to further refine the model for performance improvements on mobile devices.
- Validation: Ensuring the converted model performs as expected by running inference tests on a representative dataset.

4.4.2 Integration into android studio application

Figure 11 provides a visual representation of the subsequent steps in our application's development. We seamlessly integrated the TensorFlow Lite model into the Android Studio application, leveraging the OpenCV library for enhanced functionality. The resulting system is designed to simplify the process of learning Arabic alphabet signs through an interactive and visually engaging experience. Steps include:

- Android Studio Setup: Configuring the development environment, including installing necessary SDKs and setting up the project structure.
- TensorFlow Lite Interpreter Integration: Adding the TensorFlow Lite dependency and writing code to load and run the model using the TensorFlow Lite Interpreter.
- OpenCV Integration: Incorporating OpenCV for image processing tasks, such as hand detection and preprocessing the input images for the model.
- User Interface Design: Creating an intuitive user interface that allows users to interact with the application easily.

4.4.3 User interaction and learning process

As illustrated in Fig. 11, our system offers a user-friendly interface for learning Arabic alphabet signs. Users select the "Arabic Alphabet Sign Recognition" option and use their device's camera to position their hand and form an Arabic sign. The system, powered by the TensorFlow Lite model, detects the hand gesture, frames it in red, and displays the corresponding Arabic character at the top of the screen. This interactive approach facilitates an immersive learning experience.

4.4.4 Real-time hand sign detection

Figure 12 showcases the practical outcomes of real-time hand sign detection using our application. The displayed results demonstrate the accurate translation of hand signs into Arabic characters. Despite potential similarities between certain signs, such as 'jeem,' 'haa,' and 'khaa,' our system excels in accurate detection. This capability

Table 7 Classification report

Alphabet	Precision (%)	Recall (%)	F1-Score (%)	Support
Ain	100.00	100.00	100.00	211
Al	1.0000	99.25	99.63	134
Aleff	99.40	100.00	99.70	167
Bb	98.90	100.00	99.44	179
Dal	98.78	99.39	99.08	163
Dha	99.40	95.95	97.65	173
Dhad	100.00	98.80	99.40	167
Fa	98.48	99.49	98.98	196
Gaaf	99.40	98.24	98.82	170
Ghain	99.49	99.49	99.49	198
Ha	99.38	100.00	99.69	159
Haa	99.35	100.00	99.67	152
Jeem	99.35	98.71	99.03	155
Kaaf	99.43	97.75	98.58	178
Khaa	100.00	100.00	100.00	160
La	99.43	100.00	99.71	174
Laam	99.45	98.91	99.18	184
Meem	100.00	99.44	99.72	177
Nun	98.90	98.90	98.90	182
Ra	98.22	100.00	99.10	166
Saad	98.95	98.95	98.95	190
Seen	100.00	99.39	99.69	164
Sheen	100.00	98.67	99.33	150
Ta	96.24	98.35	97.28	182
Taa	100.00	98.91	99.45	184
Thaa	96.20	100.00	98.06	177
Thal	100.00	98.10	99.04	158
Toot	99.44	100.00	99.72	179
Waw	99.28	100.00	99.64	137
Ya	100.00	100.00	100.00	172
Yaa	100.00	100.00	100.00	129
Zay	99.28	99.28	99.28	138

Table 8 Metric report

Test Accuracy (%)	99.24
Macro Avg	
Precision (%)	99.27
Recall (%)	99.25
F1-score (%)	99.26
Weighted Avg	
Precision (%)	99.25
Recall (%)	99.24
F1-score (%)	99.24

provides learners with a rapid and effective method for acquiring proficiency in Arabic alphabet signs.

The efficiency of our approach is attributed to the exceptional performance of the EfficientNet B7 architecture compared to other evaluated architectures. Individuals facing challenges in recognizing and assimilating Arabic alphabet signs stand to benefit significantly from our application, thanks to its precision and efficiency.

5 Limitations and challenges

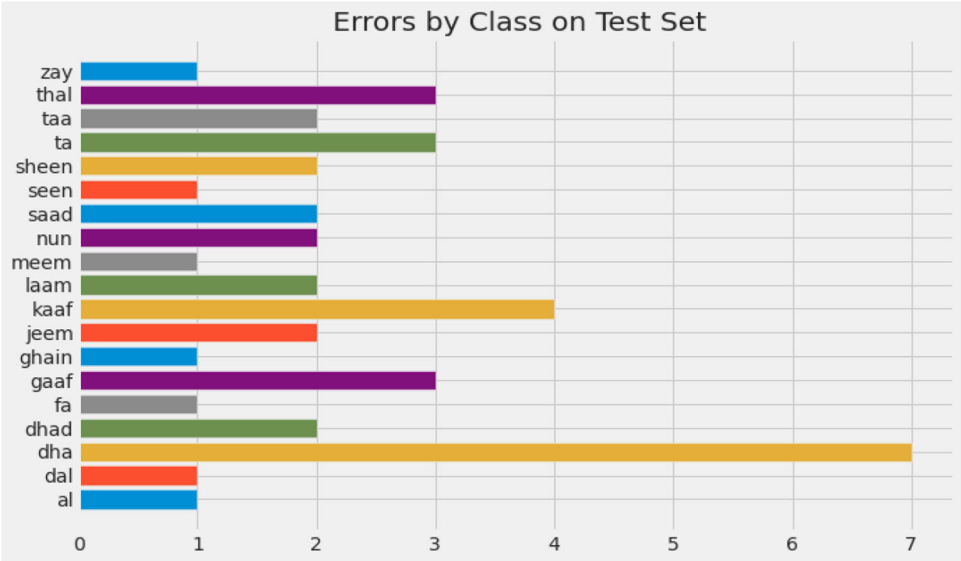
Despite the satisfactory performance of our system, as demonstrated in Fig. 14, there are inherent limitations and challenges that affect real-time sign identification. These issues stem from variations in brightness and changes in the background, which can hinder the system's ability to detect signs promptly. It is essential to note that the EfficientNet B7 model, trained on greyscale images with simple backgrounds from the ArASL2018 database, exhibits detection errors when confronted with background alterations. In instances where the background remains simple, the system operates optimally, but it may fail to detect signs in more dynamic environments.

Figure 13 shows examples of Arabic signs that were not successfully detected and recognized by our system. Notably, these failures occurred under changing background conditions, highlighting the vulnerability of the system in such scenarios. Among the causes were the printed background and the level of brightness. Sometimes our hand was not even framed, i.e., detected, and other times the frame existed but did not detect the region of interest (the hand), resulting in poor recognition.

While our results are promising, addressing these challenges is crucial for enhancing the real-time recognition rate of our system. In light of these limitations, we have outlined plans for further research to overcome these obstacles. Ongoing efforts focus on refining the model to adapt to varying backgrounds and brightness levels, ensuring robust performance in diverse settings. While our results are promising, addressing these challenges is crucial for enhancing the real-time recognition rate of our system. In light of these limitations, we have outlined plans for further research to overcome these obstacles. Ongoing efforts focus on refining the model to adapt to varying backgrounds and brightness levels, ensuring robust performance in diverse settings.

We introduce a new comprehensive database, ArASL +, encompassing 28 classes of Arabic letter signs. This database consists of 90 color photos per class, each measuring 64 × 64 pixels, as shown in Fig. 14. The images are captured from various angles, under different brightness levels, and against complex backgrounds.

Fig. 9 Errors by class on test set



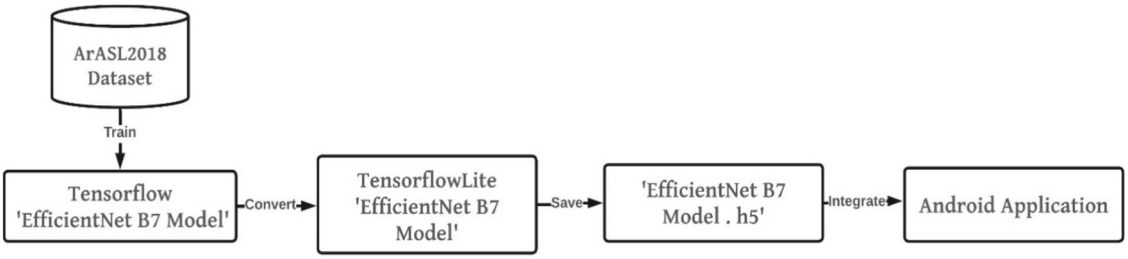


Fig. 10 Guidelines for Incorporating a TensorFlow Model into an Android Application

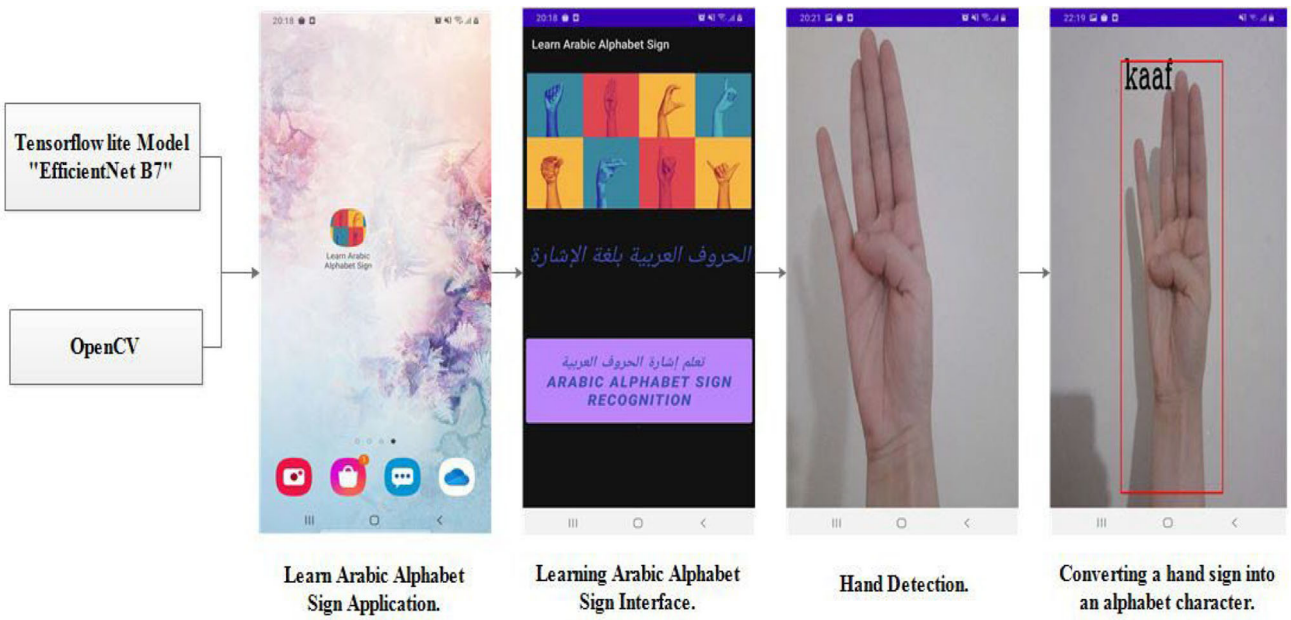


Fig. 11 The procedures for recognizing the signs of the alphabet

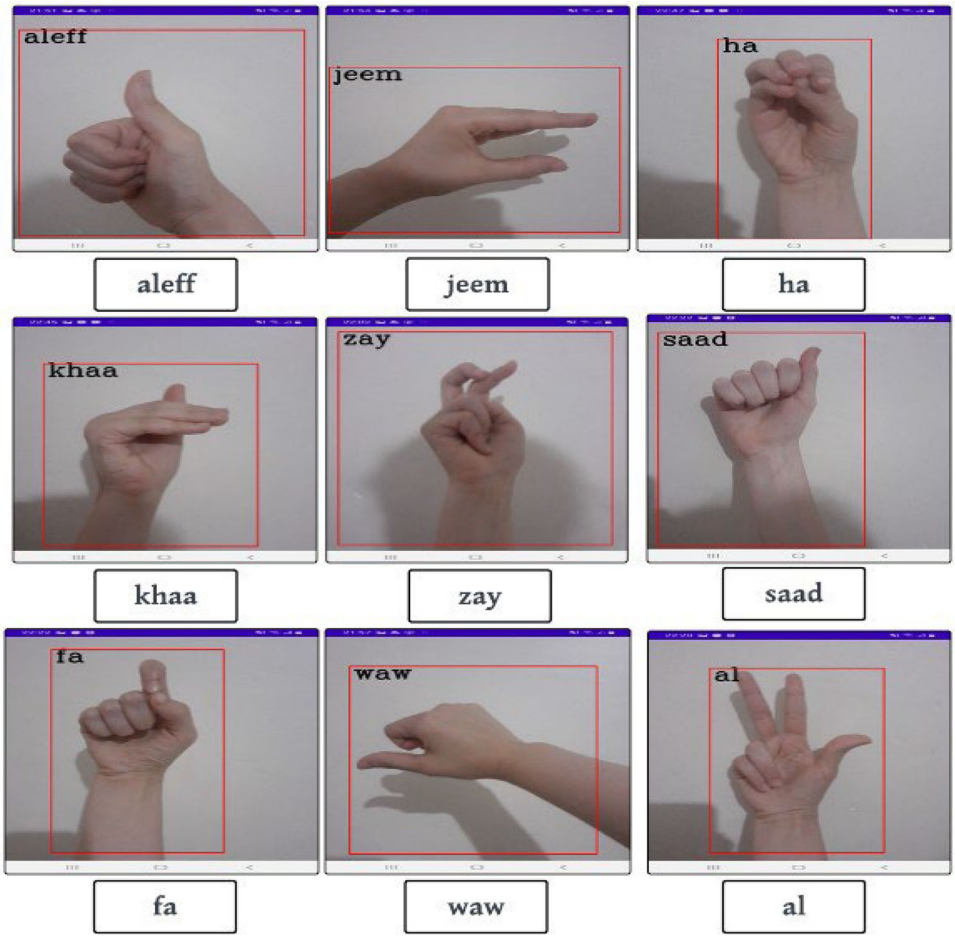


Fig. 12 Examples of Arabic Signs Recognition

Fig. 13 Examples of Arabic signs failed detection and recognition

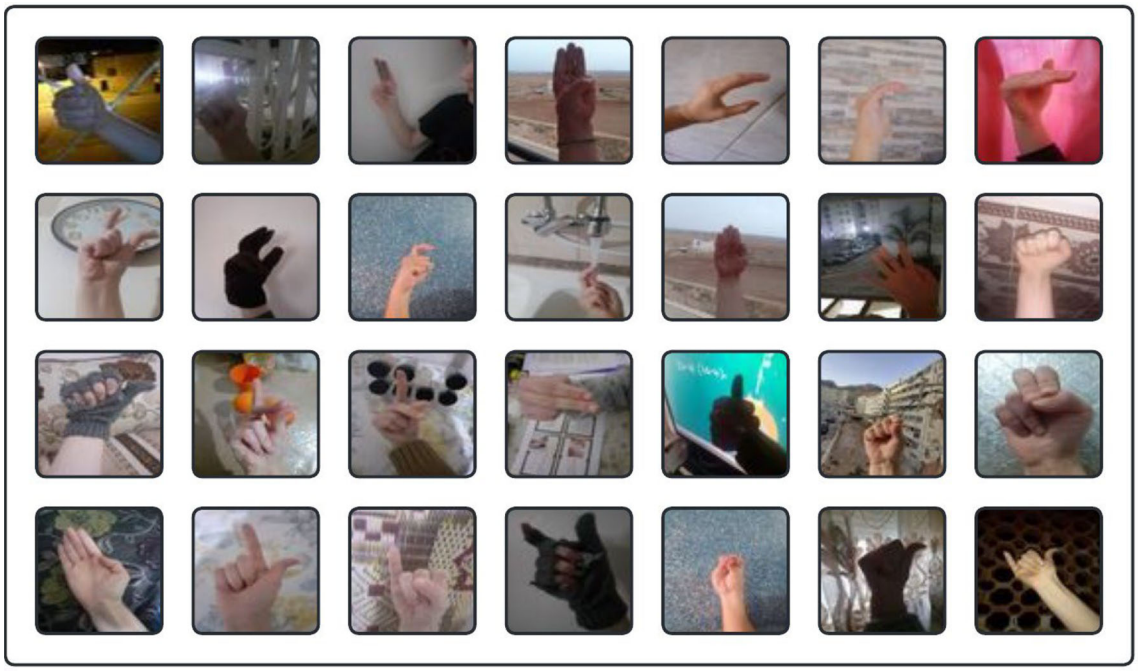


Fig. 14 ArASL + dataset

ArASL + is freely accessible on Kaggle, serving as a valuable resource to foster research and facilitate the development of more resilient solutions in the domain of Arabic sign recognition. The database can be accessed at: https://www.kaggle.com/datasets/sarra1350/arasl

6 Conclusion

In conclusion, our work centers on the development of an educational application tailored for acquiring proficiency in the Arabic sign alphabet, addressing the unique requirements of deaf individuals within the Arab community. This endeavor leverages the ArASL2018 dataset, to provide a resource that aligns with the specific linguistic and cultural nuances of Arabic sign language.

Our investigation encompassed the thorough presentation, evaluation, and analysis of twelve Convolutional Neural Network (CNN) classification models, all implemented on a computer processor. The overarching objective was to discern the optimal model for the intended application. Through meticulous experimentation, the results unequivocally endorsed the EfficientNet B7 model—a pre-trained CNN architecture—as the standout performer. This model showcased remarkable prowess, achieving an impressive accuracy of 99.24% on test data.

Furthermore, the application's real-time capabilities stand as a noteworthy advantage, facilitating instantaneous feedback and interaction for users. The EfficientNet B7 model's robust performance ensures swift and accurate recognition of Arabic sign alphabet gestures, enhancing the learning experience. This real-time functionality not only expedites the learning process but also fosters a dynamic and engaging educational environment, crucial for effective language acquisition.

Funding The interest statement does not apply to the manuscript. No funding was received to assist with the preparation of this manuscript.

Data Availability Statement The interest statement does not apply to the manuscript. No funding was received to assist with the preparation of this manuscript. This study utilized the publicly available ArASL2018 dataset, which can be accessed at Mendeley Data (https://data.mendeley.com/datasets/y7pckrw6z2/1), (DOI: https://doi.org/10.17632/y7pckrw6z2.1). Additionally, we created a new dataset, ArASL+, which is freely available at Kaggle (https://www.kaggle.com/datasets/sarra1350/arasl). Alternatively, it can be obtained by contacting the corresponding author.

Declarations

Conflict of interest (a) All authors have participated in conception and design, or analysis and interpretation of the data. (b) Drafting the article or revising it critically for important intellectual content. (c) Approval of the final version. (d) This article includes the generation and analysis of a new dataset, the ArASL + database,

References

Ahmed S, Bons M (2020) Edge computed NILM: a phone-based implementation using MobileNet compressed by tensorflow lite.

In: Proceedings of the 5th International Workshop on non-intrusive load monitoring, pp 44–48
Akbar AS, Fatichah C, Suciati N (2020) Modified MobileNet for patient survival prediction. International MICCAI Brainlesion Workshop. Springer International Publishing, Cham, pp 374–387
Ali K, Shaikh ZA, Khan AA, Laghari AA (2022) Multiclass skin cancer classification using EfficientNets–a first step towards preventing skin cancer. Neurosci Inform 2(4):100034
Almasre MA, Al-Nuaim H (2016) A real-time letter recognition model for Arabic sign language using kinect and leap motion controller v2. Int J Adv Eng Manag Sci 2(5):239469
Alsaadi Z, Alshamani E, Alrehaili M, Alrashdi AAD, Albelwi S, Elfaki AO (2022) A real time Arabic sign language alphabets (ArSLA) recognition model using deep learning architecture. Computers 11(5):78–98. https://doi.org/10.3390/computers11050078
Barbu A, Mayo D, Alverio J, Luo W, Wang C, Gutfreund D, Katz B (2019) Objectnet: a large-scale bias-controlled dataset for pushing the limits of object recognition models. In: Advances in neural information processing systems 32 (NeurIPS 2019)
Belissen V, Braffort A, Gouiffès M (2020) Dicta-Sign-LSF-v2: remake of a continuous French sign language dialogue corpus and a first baseline for automatic sign language processing. In: LREC 2020, 12th Conference on Language Resources and Evaluation
Bird JJ, Ekárt A, Faria DR (2020) British sign language recognition via late fusion of computer vision and leap motion with transfer learning to american sign language. Sensors 20(18):5151
Cai K, Miao X, Wang W, Pang H, Liu Y, Song J (2020) A modified YOLOv3 model for fish detection based on MobileNetv1 as backbone. Aquacult Eng 91:102117
Cenggoro, T. W. (2020). Incorporating the knowledge distillation to improve the efficientnet transfer learning capability. In *2020 International Conference on Data Science and its Applications (ICoDSA)* (pp. 1–5). IEEE.
Chen MC, Ball RL, Yang L, Moradzadeh N, Chapman BE, Larson DB, Lungren MP (2018) Deep learning to classify radiology free-text reports. Radiology 286(3):845–852
Chen X, Li Y, Hu R et al (2020) Hand gesture recognition based on surface electromyography using convolutional neural network with transfer learning method. IEEE J Biomed Health Inform 25(4):1292–1304
Chicco D, Jurman G (2020) The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genom 21(1):1–13
Chicco D, Tötsch N, Jurman G (2021) The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. BioData Min 14(1):1–22
Csizér K, Kontra EH (2020) Foreign language learning characteristics of deaf and severely hard-of-hearing students. Mod Lang J 104(1):233–249
Cuxac C (2001) Les langues des signes: analyseurs de la faculté de langage. Acquisition et Interaction en Langue Étrangère (15):11–36. https://doi.org/10.4000/aile.536
Deriche M, Aliyu SO, Mohandes M (2019) An intelligent arabic sign language recognition system using a pair of LMCs with GMM based classification. IEEE Sens J 19(18):8067–8078
Duarte A, Palaskar S, Ventura L, Ghadiyaram D, DeHaan K, Metze F, Giro-i-Nieto X (2021) How2sign: a large-scale multimodal dataset for continuous American sign language. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 2735–2744)
Edel G, Kapustin V (2022) Exploring of the MobileNet V1 and MobileNet V2 models on NVIDIA Jetson Nano microcomputer. J Phys Conf Ser 2291(1):012008
Elatawy SM, Hawa DM, Ewees AA, Saad AM (2020) Recognition system for alphabet Arabic sign language using neutrosophic and fuzzy c-means. Educ Inf Technol 25:5601–5616
El-Bendary N, Zawbaa HM, Daoud MS, Hassanien AE, Nakamatsu K (2010) Arslat: Arabic sign language alphabets translator. In: 2010 International Conference on computer information systems and industrial management applications (CISIM), pp 590–595. IEEE
Fadlilah U, Handaga B (2021) The development of android for Indonesian sign language using tensorflow lite and CNN: an initial study. J Phys Conf Ser 1858(1):012085
Flach P, Kull M (2015) Precision-recall-gain curves: PR analysis done right. In: Advances in neural information processing systems (NIPS 2015), vol 28, pp 838–846. https://dl.acm.org/doi/10.5555/2969239.2969333
Fleurion D, Verdun S, Ridoux I, Scemama C, Bouillevaux I, Ciosi A, Drion B (2021) Transposition and normalization of the mini-mental state examination in French sign language. Arch Clin Neuropsychol 36(6):990–1002
Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
Guan Q, Wang Y, Ping B, Li D, Du J, Qin Y, Xiang J (2019) Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study. J Cancer 10(20):4876
Han S, Jeong J (2020) An weighted CNN ensemble model with small amount of data for bearing fault diagnosis. Proc Comput Sci 175:88–95
Han X, Zhang Z, Ding N, Gu Y, Liu X, Huo Y, Zhu J (2021) Pre-trained models: PAST, present and future. AI Open 2:225–250
Hoang VT, Jo KH (2021) Practical analysis on architecture of EfficientNet. In: 2021 14th International Conference on Human System Interaction (HSI), pp 1–4. IEEE.
Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
Jiang X, Satapathy SC, Yang L et al (2020a) A survey on artificial intelligence in Chinese sign language recognition. Arab J Sci Eng 45(12):9859–9894
Jiang X, Lu M, Wang SH (2020b) An eight-layer convolutional neural network with stochastic pooling, batch normalization and dropout for fingerspelling recognition of Chinese sign language. Multimedia Tools Appli 79:15697–15715
Kallipolitis A, Revelos K, Maglogiannis I (2021) Ensembling EfficientNets for the classification and interpretation of histopathology images. Algorithms 14(10):278
Kamruzzaman MM (2020) Arabic sign language recognition and generating Arabic speech using convolutional neural network. Wirel Commun Mobile Comput 2020(1):3685614
Khalifa NE, Loey M, Mirjalili S (2022) A comprehensive survey of recent trends in deep learning for digital images augmentation. Artif Intell Rev 55(3):2351–2377. https://doi.org/10.1007/s10462-021-10066-4
Koonce B, Koonce BE (2021) EfficientNet. In: Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization. Apress, New York, NY, USA, pp 109–123. https://doi.org/10.1007/978-1-4842-6168-2
Latif G, Mohammad N, Alghazo J, AlKhalaf R, AlKhalaf R (2019) ArASL: Arabic alphabets sign language dataset. Data Brief 23:103777
Lincy RB, Gayathri R (2020) Off-Line Tamil handwritten character recognition based on convolutional neural network with VGG16 and VGG19 model. In: International Conference on Automation, signal processing, instrumentation and control, pp 1935–1945. Singapore: Springer Nature Singapore.

Marques G, Agarwal D, De la Torre Díez I (2020) Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. Appl Soft Comput 96:106691

Mohandes M, Deriche M (2005) Image based Arabic sign language recognition. In: 8th International Symposium on Signal Processing and its Applications, ISSPA 2005, pp 86–89

Mulim W, Revikasha MF, Hanafiah N (2021) Waste classification using EfficientNet-B0. In: 2021 1st International Conference on computer science and artificial intelligence (ICCSAI), Vol. 1, pp 253–257. IEEE

Mustafa M (2021) A study on Arabic sign language recognition for differently abled using advanced machine learning classifiers. J Ambient Intell Humaniz Comput 12(3):4101–4115. https://doi.org/10.1007/s12652-020-01790-w

Naglot D, Kulkarni M (2016) Real time sign language recognition using the leap motion controller. In: 2016 International Conference on inventive computation technologies (ICICT), Vol. 3, pp. 1–5. IEEE

Ng CW, Lee KM, Tang DK (2004) Three-dimensional numerical investigations of new Austrian tunnelling method (NATM) twin tunnel interactions. Can Geotech J 41(3):523–539

Organization WH, et al (2019) Deafness and hearing loss [www document]. URL https://www.who. int/news-room/fact-sheets/detail/deafness-and-hearing-loss. Accessed 9 Aug 2019) (2019)

Pratama ATM, Pratama AR (2021) Rancang Bangun Aplikasi Android "Kuliah Apa?" Berbasis Flutter dan TensorFlow Lite. Automata 2(1)

Qassim H, Verma A, Feinzimer D (2018) Compressed residual-VGG16 CNN model for big data places image recognition. In: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), pp 169–175. IEEE

Qiu X, Sun T, Xu Y, Shao Y, Dai N, Huang X (2020) Pre-trained models for natural language processing: a survey. SCIENCE CHINA Technol Sci 63(10):1872–1897

Rahman MM, Islam MS, Rahman MH, Sassi R, Rivolta MW, Aktaruzzaman M (2019) A new benchmark on American sign language recognition using convolutional neural network. In: 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), pp. 1–6. IEEE

Rawf KMH, Mohammed AA, Abdulrahman AO, Abdalla PA, Ghafoor KJ (2022) A comparative technique using 2D CNN and transfer learning to detect and classify Arabic-Script-based sign language. Acta Inform Malaysia 7(1):8–14

Rhee YG, Cho NS, Lim CT, Yi JW, Vishvanathan T (2008) Bridging the gap in immobile massive rotator cuff tears: augmentation using the tenotomized biceps. Am J Sports Med 36(8):1511–1518

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018).Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 4510–4520

Shin J, Matsuoka A, Hasan MAM, Srizon AY (2021) American sign language alphabet recognition by extracting feature from hand pose estimation. Sensors 21(17):5856

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sutton-Spence R, Woll B (2004) British sign language. In: Davies A, Elder C (eds) The handbook of applied linguistics. Blackwell Publishing Ltd., pp 165–186. https://doi.org/10.1002/9780470757000

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on machine learning, pp 6105–6114. PMLR

Tan M, Le Q (2021) Efficientnetv2: smaller models and faster training. In: International Conference on machine learning, pp 10096–10106. PMLR

Theckedath D, Sedamkar RR (2020) Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. SN Comput Sci 1:1–7

Wanjaya I, Goncharenko I, Gu Y (2022) Comparison of image-based and skeleton-based ML methods in the task of alphabetical sign language recognition. In: 2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech), pp. 316–317. IEEE

Yacouby R, Axman D (2020) Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In: Proceedings of the First Workshop on evaluation and comparison of NLP systems, pp 79–91

Yi SL, Yang XL, Wang TW, She FR, Xiong X, He JF (2021) Diabetic retinopathy diagnosis based on RA-EfficientNet. Appl Sci 11(22):11035

Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, He Q (2020) A comprehensive survey on transfer learning. Proc IEEE 109(1):43–76