Balkan Journal of Electrical and Computer Engineering (BAJECE)
Volume 1 Number 1, September 2023

A Real-time Face Recognition Based on MobileNetV2 Model

Vafaa Sukkar[1] and Ergun Ercelebi[2]

1,2 Department of Electrical and Electronics Engineering, University of Gaziantep, 27310 Gaziantep, Turkey

Manuscript received Jan 22, 2023; accepted August 12, 2023.

***Abstract*—Facial recognition technology is one of the fastest developing technologies. It is the most widespread technology compared to other biometric ones. Technologies based on deep learning and neural networks have demonstrated superior efficiency and speed when compared to traditional approaches for recognizing persons. In this work, we introduce a light system for real-time facial recognition with an improved recognition time. It relies on today's latest convolution neural networks algorithms. The database is built based on photos from a collection of both known people and some VggFace dataset celebrities. The system pipeline is divided into several phases, beginning with detecting faces in input images using the MTCNN algorithm, then aligning and preprocessing them, extracting facial characteristics vectors for each face using a pre-trained mobileNetV2 model, and finally classifying faces using the SoftMax classifier layer. Evaluating its performance on some samples, the system achieved an average accuracy of 92.67% with an execution time of 10 milliseconds to process an image.**

Index Terms—**Deep Learning, Face Recognition, MobileNetV2, MTCNN, Real-time**

I. Introduction

Face recognition technology has sparked much attention as an effective and secret tool that provides a doorway into massive amounts of data and enables the identification of people without their feeling or knowledge.

Because of its simplicity, low cost, flexibility of use, and immediate results providence, facial recognition is the most appropriate way in the majority of cases to identify people compared to the prevailing methods, such as fingerprints, which require that the person's finger be clean and place it on the scanner, or as DNA that requires samples to be analyzed and this it takes time and money, or such as the eye print that requires being very close to the reading camera, or the voice print that requires shouting sometimes and approaching the microphone, or the RF technology that the person who may already doubt his credibility has to take the RFID card out of his pocket and touch the surfaces that may be contaminated.

Often there is confusion between facial identification and facial authentication, but they are different techniques and usually designed for different purposes. Facial authentication or verification systems verify the person by matching his photo with a specific identity. We see such systems in mobile phones that are being unlocked after checking and verifying the image of the owner's face. It is a 1:1 matching process. Face identification systems identify a person by comparing the image of his face with the pre-defined people in the database and determining the identity of this person. An example of this is the system in some universities; once the student arrives at the university, the system recognizes him/her after analyzing his/her facial image, and the gate automatically opens then. Face identification is a 1: N process.

The pipeline of facial recognition systems is generally composed of sequential steps that help capture, analyze, compare, and match the captured face to a precompiled database of images. Face Detection is the first stage where the image is scanned and the face is distinguished from the rest of the existing objects. Followed by Alignment stage where facial landmarks are detected and localized precisely and the face is normalized to be homogenous with the dataset. The following stage is Feature Extraction in which the face is analyzed and unique data is extracted and converted into vectors for comparison. Classification is the final stage where the final extracted data is compared with the stored database of pre-defined people, Fig. 1.

Fig.1. Face Recognition building blocks.

II. Literature Review

A. Face Recognition Methods

Face detection is a kind of object detection and the primary step in face recognition systems. It is used to determine the existence of human faces and returns their location and sizes in the images if they are present, and then detect and mark each of them with a bounding box. Face Detection identifies the required components of the image for creating a faceprint. Algorithms of face detection fall under four main categories:

1) Knowledge-Based Methods (Rule-Based Methods)

In these methods, human faces are described based on rules related to the structure of the human face, as well as the

relationship and arrangement between the facial characteristics in a typical human face.

2) *Template Matching Methods*
These approaches compare input images to predefined stored template images by finding similar features and correlating the two to detect the presence of a face based on correlation values. These methods have problems with pose, shape, and scale variations.

3) *Appearance-Based Methods*
These methods are based on the templates learned from the training images that capture the representative variability of the face appearance.

4) *Feature-Based Methods*
They locate faces by searching for invariant structural features of the face and extracting them. These algorithms work even with the angle and pose variations.

B. Face Recognition Methods

The facial image is obtained from the input image and converted into numerical expression. After that, the geometry of the face is scanned by a facial recognition algorithm to dig up particular distinctive details and identify the key points of the face. The extracted data of a certain face is called a face template. Face templates are used to distinguish faces from each other by calculating and comparing the distances between the data of input and stored faces.

1) *Local features approach*
It is a part-based approach that focuses on extracting the local features in the face like mouth, nose, and eyes and determining their locations. The histogram of oriented gradients (HOG)[1], Local Binary Pattern (LBP) [2,3] and Scale Invariant Feature Transform (SIFT)[4] algorithms fall under this field. These techniques partially overcome the variation in illumination, pose, and facial expression. On the other hand, they deal with high dimensional feature space, leading to computational complexity.

2) *Holistic approach*
It deals with the entire face. The image face is described by feature vectors that are converted from the matrix of pixels representing the global information and characteristic features of faces. Dimension reduction is the key benefit of this approach, but there is a stability issue with rotations and translation cases. Independent Component Analysis (ICA)[5], Eigenfaces[6] and Principal Component Analysis (PCA) [7] are the most common techniques of this category.

3) *Hybrid approach*
It is a combination of both local features and holistic algorithms. Methods in this category utilize the strengths of the local feature approach in overcoming problems of recognition and the holistic approach in the reduction of dimensionality and complexity.

Although good results are often achieved under standard conditions, the identification of people is sometimes mistaken. This is due to several reasons, including blurred images, lack of proper illumination, different facial expressions, and the position of the face in the images. All this adversely reflects on the performance and efficiency of algorithms and prevents proper analysis and precise results, especially in real-time recognition.

C. Deep Learning and Convolutional Neural Network

Facial recognition techniques have undergone great changes and evolutions throughout the years, especially in the last decade, resulting in a rapid expansion of use in commercial applications. Classical face recognition techniques focus on using geometry-based methods and statistical subspaces. However, due to variations caused by view angles, background clutter, and occlusions, these methods reflect some failures in representing faces. Thanks to deep learning, facial recognition has become more powerful, and less affected by varying conditions.

Deep learning (DL) is a categorization of the Machine Learning classification that falls under Artificial Intelligence. It is centered on the creation of algorithms and models that can learn and make intelligent judgments without human intervention. A deep learning system is a self-learning system that relies primarily on deep neural networks and learns through passing, processing, and filtering data within the networks. Convolutional neural network (ConvNet, or CNN) is one of the deep learning methods. They are similar in shape to artificial neural networks (ANN), which are the backbone of deep learning, as they are hierarchical. CNN, like ANN, has learnable weights and biases. What distinguishes CNN is that it is primarily utilized for images classification and facial recognition issues, with an image as the input in this case. The CNN transforms the image into a simpler form with fewer dimensions without losing its properties, making it easier to analyze and process.

The ability to train CNN-based models with vast data sets to learn the best data representation features is the primary benefit of the deep learning-based approaches. Also, the availability of large data sets of diverse faces images of people has contributed to the superiority of these methods.

Convolutional neural networks[8] consist of several layers:

1) *Input layer*
The input image is converted into a matrix of numbers that represent its pixels.

2) *Convolutional layers*
These layers extract low-level and high-level features of the image in phases until all the characteristics are extracted. The first convolutional layer could be confined to extracting low-level features such as colors and edges.

3) *Pooling layer*
This layer sits between the convolutional and FC layers to reduce the spatial size of the image and hence decrease the computational cost. A pooling layer might be one of two kinds: average pooling or max pooling.

4) *Fully connected layers*
These layers usually represent the last layers of CNN. The input to these layers is the output of the last layer of the convolution layer or pooling output (if it exists) after it is flattened. Through training, a fully connected layer collects extracted data from previous layers and feeds them into the Softmax layer, which is the classification layer, Fig. 2.

With the spread and popularity of deep learning methods, researches on facial recognition accelerated, and CNNs are used to deal with many other issues such as objects detection, handwritten character recognition, translations, question answering, analysis of facial expressions, and others.

Deep learning-based methods are now the most successful among facial recognition techniques; they provide the best results compared to all other algorithms, especially with the significant development in convolutional neural network architectures such as R-CNN, Fast R-CNN, VGG16 and ResNet50. Despite these algorithms being among the most commonly used algorithms for face recognition, they have main problem related to poor processing speed which make them unapplicable in real-time cases or problems that require fast outputs.

D. *Related Work*

Authors in research[9] proposed a robust face recognition system based on CNN. Viola-Jones algorithm was used for detecting the face. Then contrast enhancement using Histogram Equalization proceeded to the input face. They implemented their work on the Extended Yale B and CMU PIE face databases. Their work achieved a recognition rate of 97.23% and 98.38% on both VGG16 and ResNet50 architectures, respectively.

Other researchers[10] designed a FR system that takes the attendance automatically using deep learning technology. The maximum recognition rate of their system was 70%. Face detection was carried out using Haar cascade method, whereas face recognition was performed using LBPH.

In research[11], the authors designed an attendance system using computer vision and machine learning technology. They used a DNN-based detector for detection and LDA and PCA methods for feature extraction. Their FR method achieved a real-time accuracy of 56% for MLP and SVM classifiers and about 89% for the CNN classifier.

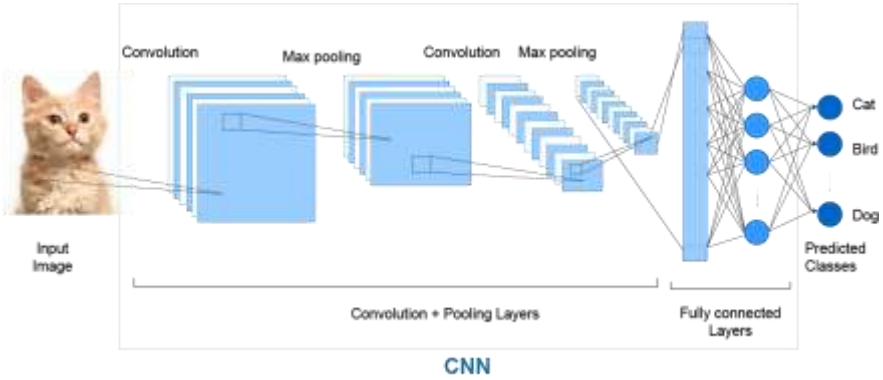

Fig.2. An example of a Convolutional Neural Network.

III. METHODOLOGY

This work aims to design a real-time facial recognition system that is robust, fast, and light at the same time using predefined CNN algorithms. The fundamental reason for this is that it is more practical to integrate a system into any type of device, whether it is an embedded system, a mobile device, or a computer without GPU, regardless of its capabilities.

The proposed face recognition system begins with the input image, where the face is detected and located using the MTCNN algorithm. After that, the image is aligned, and the face is cropped from it. The deep CNN model MobileNetV2 is applied to extract features from the cropped face, and the classification process is performed using the SoftMax layer classifier.

A. *Face Detection*

We used the Multi-task Cascaded Convolutional Neural Network (MTCNN) algorithm to efficiently search for faces in the image, detect them and recognize their facial marks such as eyes, lips, eyebrows, etc. MTCNN is a robust algorithm presented to perform both face detection and alignment. It detects faces with high speed and accuracy, and it is more potent than other algorithms in encountering the challenges that negatively affect the detector's efficiency, including the conditions in which the image was taken and changes in the face.

MTCNN is made up of three separate cascade stages of networks. They are P-Net, R-Net, and the O-Net. The output of one stage is the input to the next stage. Each of these networks returns three information: a face bounding rectangle, the probability that a particular rectangle contains a face, and five landmarks.

For the detector to be able to detect faces of all sizes, copies of the image at different scales are created as a first step, resulting in an image pyramid.

The overall three stages of MTCNN, Fig. 4, are as follow:

1) Proposal network (P-Net)

It is a Fully Convolutional Network (FCN) utilized to fast analyze the image and returns many candidate windows with corresponding boundary box regression vectors. These are then filtered using the Non-Maximum Suppression (NMS) technique to downsize the candidate windows and obtain the best boundary boxes out of the overlapping boundary boxes.

2) Refine Network (R-Net)

It is a CNN since the dense layer exists in the architecture of this network. The network further filters the predicted candidate windows from the previous P-Net and provides more credible and accurate boundary boxes accomplished with the confidence level of each of them. The NMS is then applied again to clear out those boundary boxes of low confidence.

3) The Output Network (O-Net)

It is more complicated CNN than R-Net, and it is the slowest network of the three cascade networks since it aims to get more facial features and returns locations of the five facial landmarks, including right and left eyes, nose, right and left corners of the

mouth. After this stage, only one boundary box should remain for each face in the image.

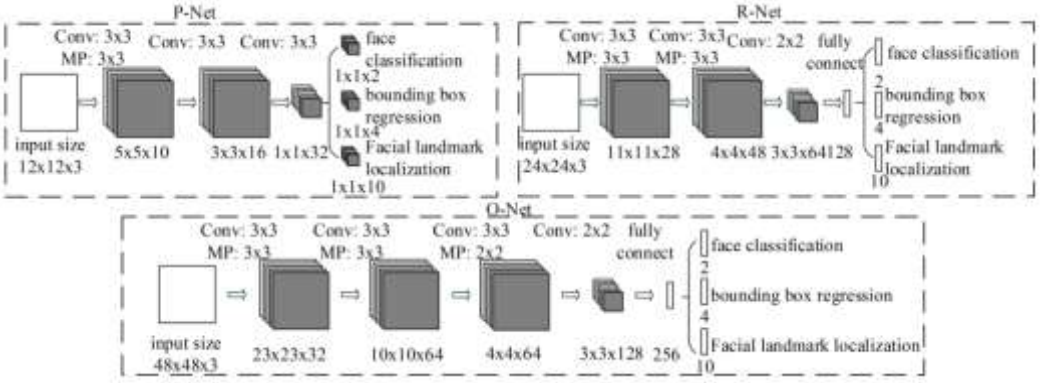

Fig.3. MTCNN stages.

B. *Face Alignment*

Facial alignment is essential as it improves the overall performance of the recognition system and provides higher accuracy. After the face has been discovered in the picture, it must be centered, Fig. 4. So, the next step is to locate the face and identify the facial markers like eyes and the nose. Fortunately, the MTCNN algorithm locates the face and its components; all that's left is to:

1. Use the coordinates of these facial components and analyze their positions to estimate the required rotation angle.
2. Rotate the face so that the eyes are at the same horizontal level.
3. Center the face in the image.
4. Cut it and change its size to fit the classification network input size.

Fig.4. Face detection and alignment.

C. *Feature Extraction and Face Classification*

MobileNetV2[12]is one of the CNN models that are used for image recognition. This algorithm has high effectiveness, performance, and speed in extracting the features. Moreover, it is light and applicable for mobile devices and devices of low computational power.

Compared to the standard CNN algorithms with similar depth, MobileNet[13] has significantly less model size and uses a smaller number of parameters. It uses Depthwise separable convolution, which conduces to less computational cost since it reduces the multiplication and addition operations. Fig. 5 and Fig. 6 show the standard convolution and the separable depthwise convolution, respectively

Depthwise separable convolution is split into two operations:

1) Depthwise Convolution

Unlike in the normal convolutions where the convolution is applied to all or multiple input channels at a time, in depthwise convolution, the convolution of a kernel is performed over a single channel. The output is then shaped by stacking the outputs of these channels.

2) Pointwise Convolution

It is a 1x1 convolution applied at each point on the M channels to change the size of the depthwise convolution output. The kernel's channels are equal to the number of input channels.

Authors of MobileNet paper show that the ratio of the total computational cost of depthwise separable convolution compared to normal convolution is: $1/N + 1/ D_k \times D_k$. When N is larger, as in normal cases, the total cost of depthwise separable convolution will be around ten times cheaper in computational cost.

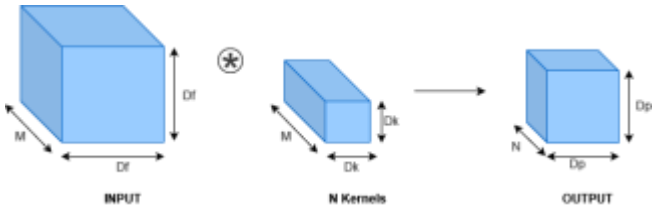

Fig.5. The standard convolution.

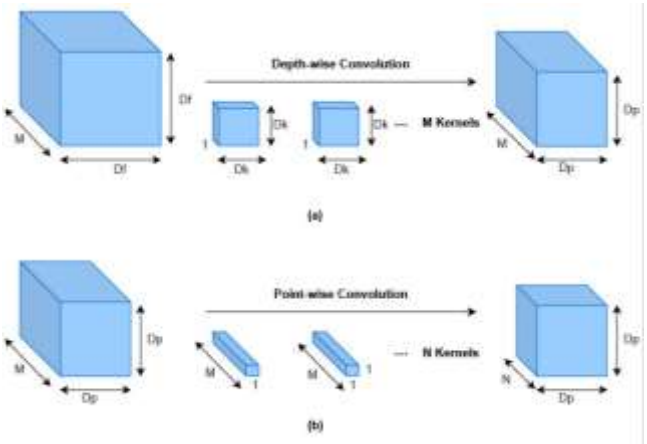

Fig.6. Separable Depthwise convolution: (a) Depthwise convolution (b) Pointwise convolution.

MobileNetV2 architecture is introduced on the basis of MobileNetV1 to increase the accuracy and reduce the computational cost more. The residual connections and the expansion layer are two new features applied to mobileNetV2

architecture. MobileNetV2 is based on two types of Bottleneck blocks. Each block has three different convolutional layers: 1x1 Convolution with Relu6, Depthwise Convolution, and 1x1 Convolution.

MobileNetV2 network is built of 53 convolutional layers and 19 blocks. Table 1 [12] shows the model structure, with conv2d denoting a standard 2D convolution layer, Bottleneck representing a bottleneck residual block, and AvgPool denoting the average pooling layer. The convolution is applied to the input firstly using 32 filters, then the feature extraction task is performed using the middle layers, and the classification is accomplished using the last convolution layer. *s* refers to the stride, *n* is the repeats, *t* is the expansion factor, and *c* is the number of the output channels.

TABLE I
ABSORBENCY FACTOR FOR 3% CNFs FILLER LAYER

Input H x W	number of Input channels	Operator	s	t	n	c
224x224	3	conv2d	2	-	1	32
112x112	32	Bottleneck	1	1	1	16
112x112	16	Bottleneck	2	6	2	24
56x56	24	Bottleneck	2	6	3	32
28x28	32	Bottleneck	2	6	4	64
14x14	64	Bottleneck	1	6	3	96
14x14	96	Bottleneck	2	6	3	160
7x7	160	Bottleneck	1	6	1	320
7x7	320	Conv2d	1	-	1	1280
7x7	1280	AvgPool	-	-	1	-
1x1	1280	conv2d	-	-	1	K

D. Transfer Learning

It is quite challenging to have that much data to train an entire CNN network from scratch. Using a model that has previously trained on a large dataset as the starting point for training on a new problem is more efficient than consuming time, effort, and power on training all of the model's layers using randomly initialized weights. The transfer learning technique is used in this face recognition system to save training time and avoid overfitting by leveraging a pre-trained model that has already learned the characteristics. The model was pre-trained on the ImageNet dataset[14], which contains more than 14 million images and over 22000 distinct categories. The model then serves as the base of our custom model for recognizing faces.

IV. EXPERIMENT AND RESULTS

Experiments were done utilizing the proposed face recognition procedure to assess the performance of the system constructed using the proposed technique. The work was split into a training stage and a testing stage. In the training stage, the mobileNetV2 model that was pre-trained on the ImageNet dataset was used as the feature extractor. The training process was performed on the classification layers newly added to the model after omitting the existing ones. To fine-tune it, we set some layers as trainable. After the model was trained and learned features, we saved it with its weights for implementation in the testing stage, Fig. 7.

A. Software and Hardware

For the implementation of face detection MTCNN and feature extraction MobileNetv2 algorithms, Python3 Programming language with TensorFlow v2.5 platform, Keras library, and Jupyter notebook were used. The training of the face recognition model was carried out on the PRO version of Google colaboratory of 12GB RAM and NVIDIA Tesla K80 GPU. The testing was then performed on a local machine: HP Laptop of Intel® Core™ i7, 2.60 GHz processor, 8.00 GB RAM, 64-bit operating system, and HP TrueVision HD Webcam with a resolution of 1280×720 (0.922MP) for real-time face recognition.

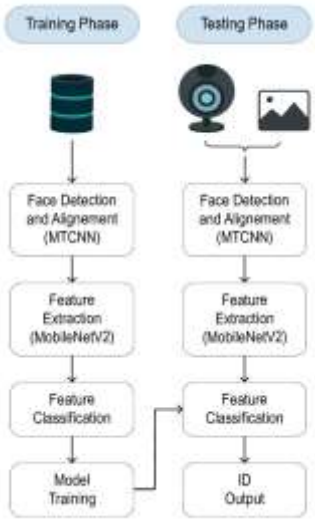

Fig.7. Block diagram of the training and testing phases.

B. Preparation of Dataset

The dataset prepared for training the system model was a mix of known people and some celebrities extracted from the VGGFace dataset. The final database consists of 1500 good-quality faces images extracted from a set of 1532 images that contain only one face. The sizes of faces extracted from the images must be at least 60 px in width and 70px in hight, and every face smaller than this size is rejected and not taken into account. The faces images belong to 15 distinct persons grouped in 15 folders labeled by the person's name. The images were taken in different conditions, with different facial expressions, poses, angles, backgrounds, and lighting conditions. The number of images for each person was fixed to 100.

C. System Implementation

The faces were detected, aligned, cropped, preprocessed, and then split into three sets; training set, validation set, and test sets with weights of 80%, 10%, and 10. MobileNetV2 model was then initialized by the pre-trained weights. For the first phase, the training process was performed on the newly added classifier layers to the model after truncating the pre-trained ones and freezing the feature extractor layers. In fine-tuning phase, some layers of the base model were unfrozen and the model was trained again using a smaller learning rate, Fig. 8.

For computing the loss, categorical_crossentropy loss function was used since we deal with a multi-class classification model. Evaluating the performance of the model was done using the accuracy function which measures the accuracy of the model and evaluates its performance. For optimization, we

employed the RMSprop (Root Mean Square Propagation) optimizer.

The total training epochs performed on the model is 57; 20 are the training stage epochs, and 37 are fine-tuning epochs. The total training epochs performed on the model is 57; 20 are the training stage epochs, and 37 are fine-tuning epochs.

Fig. 9 shows the plots of model accuracy and loss over epochs of both training and fine-tuning stages, and they are separated by a green line, and Table 2 summarizes the results of the stages.

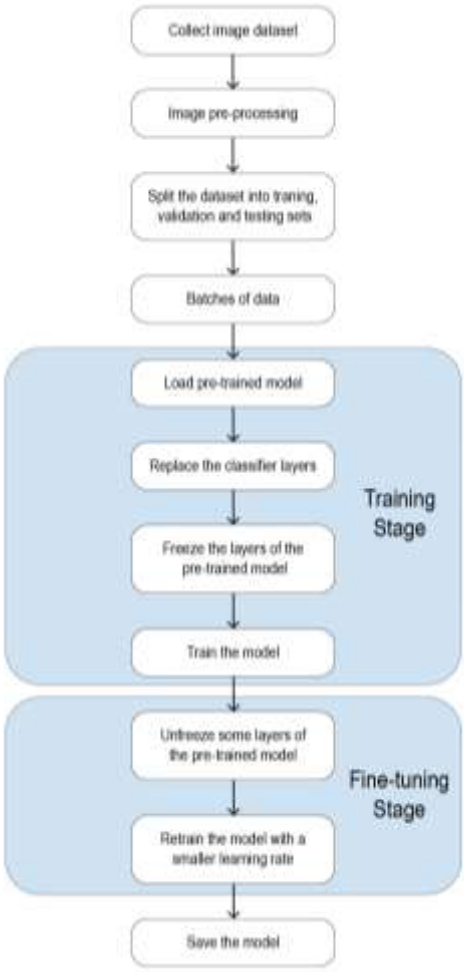

Fig.8. Proposed training work.

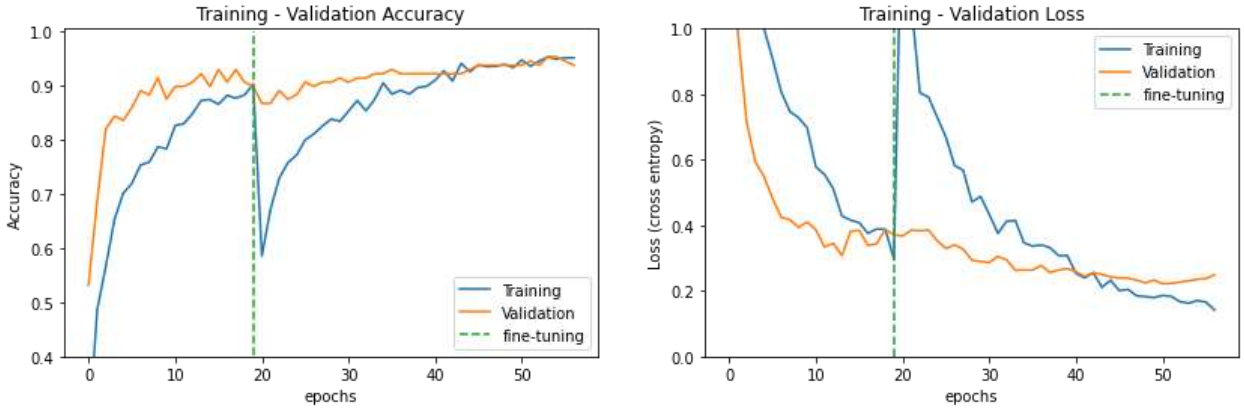

Fig.9. Model Accuracy and Loss after training and fine-tuning process.

TABLE II
Summary of Training and Fine-tuning Stages of MobileNetV2 model

		Training stage	Fine-tuning stage
Epochs		40	40
Actual epochs		20	37
Trainable parameters		32,383,503	34,244,943
Non-trainable parameters		2,257,984	396,544
Training set	Accuracy	90.24%	95.12%
	Loss	0.2980	0.1414
Validation set	Accuracy	89.84%	93.75%
	Loss	0.34	0.2492
Testing set	Accuracy	90.67%	92.67%
	Loss	0.32	0.25

The metrics used to evaluate the performance of the trained model on the testing dataset are:

$$\text{Precision(PRE)} = \frac{TP}{TP + FP}$$

$$\text{Recall(RE)} = \frac{TP}{TP + FN}$$

$$\text{F1} - \text{Score(F1)} = \frac{2 * PRE * RE}{PRE + RE}$$

$$\text{Accuracy(AC)} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP is true positive, FP is false positive, TN is true negative and FN is the false negative.

The summary of prediction outcomes on our classification model is displayed in a confusion matrix that represents the number of correct and incorrect predictions in a counted way and separates them by each class, as illustrated in fig. 10.

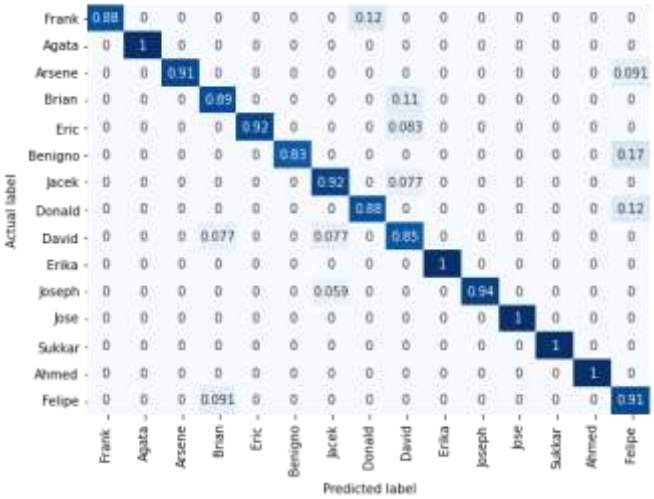

Fig.10. Normalized confusion matrix for our classification problem.

Fig. 11 displays some correct and incorrect predicted images where the incorrect prediction label is presented in red.

Fig.11. Some predicted images of the testing dataset.

D. System Testing in Real-time

The system received the loaded picture or video frame as input. Each face of the input image was detected by the MTCNN algorithm, resized to the specific size of 224x224 pixels to fit the input of MobileNetV2 model. The model.predict() function mapped and predicted the test data based on the learned labels by feeding the array into the model, determining the prediction value, and returning the label with the highest probability. The prediction value refers to the similarity ratio between the input face and the persons in the dataset. The person with the max prediction, which must be above the threshold of 85%, is the most likely to be the true identity. If the max prediction is less than that threshold, the input person image is considered as unknown and belongs to no one. Fig.12 depicts a flowchart for the real-time. And Fig. 13 is a screenshot from our system output when tested in real-time.

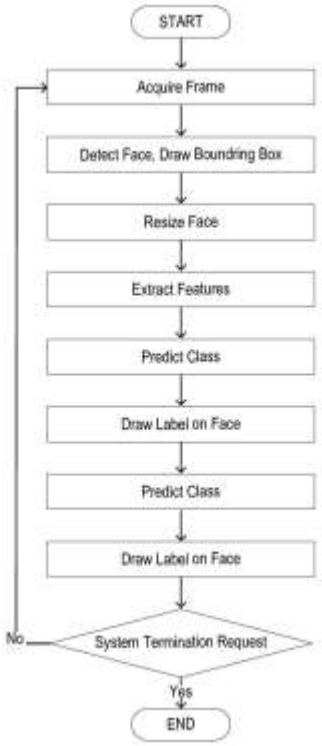

Fig.12. Flowchart of the Real-time System.

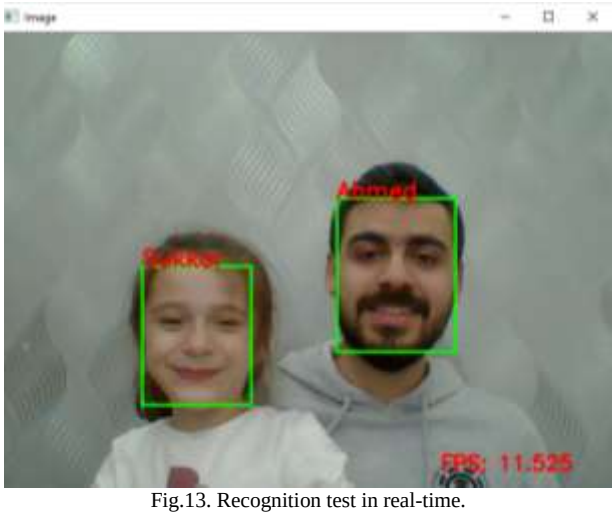

Fig.13. Recognition test in real-time.

To assess and evaluate the effectiveness of our proposed system, we compared it to VGG16 and ResNet-50 algorithms by training them on the same dataset. Using the exact procedure followed for MobileNetV2, we trained and fine-tuned the selected models. The summaries of training processes of

VGG16 and ResNet50 models are shown in tables 3 and 4, respectively.

TABLE III
Summary of Training and Fine-tuning Stages of VGG16 model

	Training stage	Fine-tuning stage
Epochs	40	40
Actual epochs	37	8
Total parameters	27,830,607	27,830,607
Trainable parameters	13,115,919	20,195,343
Non-trainable parameters	14,714,688	7,635,264
Output Accuracy	97.33%	98.67%
Output Loss	0.08	0.04

TABLE IV
Summary of Training and Fine-tuning Stages of ResNet50 model

	Training stage	Fine-tuning stage
Epochs	40	40
Actual epochs	39	27
Total parameters	75,238,799	75,238,799
Trainable parameters	51,651,087	68,866,575
Non-trainable parameters	23,587,712	6,372,224
Output Accuracy	94.00%	97.33%
Output Loss	0.023	0.09

The time it took to train and fine-tune the models were close to each other because the training processes were carried out on GPU. Resnet50 and Vgg16 models achieved higher accuracy and less loss than MobileNetV2 model. However, all the algorithms achieved more than 90% accuracy in both training and testing stages. In terms of speed and fps, MobileNetV2 surpassed the other models; it was about two times faster than Resnet-50 and three times faster than VGG16. Due to their large sizes and number of parameters, VGG16 and Resnet-50 took a long time to analyze and recognize faces, table 5.

TABLE V
Summary of Models performance in terms of execution time and fps

Model	Execution time	fps
Vgg16	0.33 s	3.011
Resnet50	0.21 s	4.707
MobileNetV2	0.09 s	11.677

V. Discussions

We trained the classifier layers first with 20 epochs and obtained an accuracy of 90.67%. Then we fine-tuned it with another 37 epochs to improve it, and we could reach 92.67% accuracy. Further increasing of epochs' number could not improve accuracy, but instead, the model could fall in overfitting. To enhance the system and avoid overfitting, we increased our dataset by applying some data augmentation on the existing images. Following face detection, we employed face alignment to rotate and center faces in pictures as needed; this stage aids in obtaining better results and a more accurate representation of extracted facial features. For speeding up the training process, we cropped all faces and extracted them from images before feeding them to the model.

Using the Fast MTCNN algorithm in real-time testing provided better results in terms of speed, where in the normal MTCNN the average fps (frames per second) was 10, whereas in Fast MTCNN, it increased four times. Previous tables reflect the robustness of our proposed work over the other analyzed models. Comparing the speed of our system with the other state-of-art models, ours recognize faces with higher fps. Considering accuracy, the results were very close to each other.

VI. Conclusion

The work presented in this paper uses a combination of two deep convolutional neural networks, MTCNN for face detection and MobileNetV2 for feature extraction, to perform a real-time facial recognition system. Using the transfer learning technique, pre-trained model weights of MobileNetV2 on the ImageNet dataset were utilized as initial values for feature extraction layers after removing its classifier and adding new classifier layers. The model then continued learning from that point on our dataset, consisting of 1500 images for 15 classes. The proposed system was tested on images and live videos and achieved an accuracy of 92.67%, with an average fps of 11.68.

References

[1] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, **I**. https://doi.org/10.1109/CVPR.2005.177

[2] Ojala, T., Pietikäinen, M., & Harwood, D. (1994). Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. *Proceedings - International Conference on Pattern Recognition*, **3**, 582–585. https://doi.org/10.1109/ICPR.1994.576366

[3] Bouwmans, T., Silva, C., Marghes, C., Zitouni, M. S., Bhaskar, H., & Frelicot, C. (2018). On the role and the importance of features for background modeling and foreground detection. *Computer Science Review*, **28**, 26–91. https://doi.org/10.1016/j.cosrev.2018.01.004

[4] Lenc, L., & Král, P. (2015). Automatic face recognition system based on the SIFT features. *Computers and Electrical Engineering*, **46**, 256–272. https://doi.org/10.1016/j.compeleceng.2015.01.014

[5] Bartlett, M. S., Movellan, J. R., & Sejnowski, T. J. (2002). Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, **13(6)**. https://doi.org/10.1109/TNN.2002.804287

[6] Turk, M. A., & Pentland, A. P. (1991). *Face recognition using eigenfaces*. doi: 10.1109/CVPR.1991.139758

[7] Sirovich, L., & Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, **4(3)**, 519. https://doi.org/10.1364/josaa.4.000519

[8] Albawi, S., Mohammed, T. A. M., & Alzawi, S. (2017). Understanding of a Convolutional Neural Network. *Ieee*, 16.

[9] Ilyas, B. R., Mohammed, B., Khaled, M., & Miloud, K. (2019). Enhanced Face Recognition System Based on Deep CNN. *Proceedings - 2019 6th International Conference on Image and Signal Processing and Their Applications, ISPA 2019*, *January*. https://doi.org/10.1109/ISPA48434.2019.8966797

[10] Harikrishnan, J., Sudarsan, A., Sadashiv, A., & Remya Ajai, A. S. (2019). Vision-Face Recognition Attendance Monitoring System for Surveillance using Deep Learning Technology and Computer Vision. *Proceedings - International Conference on Vision Towards Emerging Trends in Communication and Networking, ViTECoN 2019*, 1–5. https://doi.org/10.1109/ViTECoN.2019.8899418

[11] Damale, R. C. (2018). *Face Recognition Based Attendance System Using Machine Learning Algorithms*. *Iciccs*, 414–419.

[12] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

[13] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. http://arxiv.org/abs/1704.04861

[14] Fei-Fei, L., Deng, J., & Li, K. (2010). ImageNet: Constructing a large-scale image database. *Journal of Vision*, **9(8)**, 1037–1037. https://doi.org/10.1167/9.8.1037