







# Face Recognition with ESP-32 CAM using Neural Networks

## A Brief Comprehensive Study

Supervised by Dr\ Amjad Bayoumy

Mahmoud, Aser  222102487, Farag, Sherif  222100822, Elshabandy, Amina  222100306,  
Abdelazeem, Ahmed  221100116, Abdulrahman, ziad  221101043, Fathi, Mohamed  221101699

**Abstract** Facial recognition on resource-limited devices, such as the ESP32-CAM, presents significant challenges due to inherent processing limitations. This paper explores the development and implementation of efficient neural network-based facial recognition systems that leverage transfer learning and optimized hardware components for real-time applications. The study evaluates a well-known deep learning architecture, CNN, adapted to face recognition both datasets (premade and custom), achieving a training accuracy of 94,6% and an testing accuracy of 94,7%, making it a more reliable solution for face recognition systems.

In parallel, an ESP32-CAM-based system was developed to incorporate real-time face recognition using both celebrity datasets and user-specific datasets. This system's architecture was designed to optimize dataset preprocessing, hardware components (e.g., o-led and TTL), and neural network training to ensure efficiency and reliability in diverse environments. The CNN-based approach demonstrated robust performance in accurately identifying faces under real-world conditions, with detailed analyses provided on the challenges faced and trends observed during implementation.

These findings highlight the potential of integrating transfer learning with architectures like CNN and optimized hardware setups to achieve real-time, reliable facial recognition on resource-constrained devices. The results point to promising directions for deploying DL applications on edge devices, reducing latency, and enabling practical applications in a variety of scenarios.

**INDEX TERMS** Face recognition, image processing, neural network, artificial intelligence.

### 1. Introduction

Face recognition and individual identification have gained significant interest, with various methods available. This involves visual-image recognition, where visual patterns are identified through the eyes and processed by the brain. Similarly, computers analyze photos or videos as matrices of pixels, identifying which pixel groups represent specific objects. In face recognition, the computer's task is to accurately identify the person to whom the face data belongs.

Face recognition is a system that involves face detection, positioning, and identity verification. It uses algorithms to find face coordinates in images or videos, and uses neural networks (CNN) to identify facial features. The process involves detecting faces and comparing them to a database. The face recognition process can be based on appearance or feature-based approaches, focusing on geometric features such as eyes, nose, eyebrows, and cheeks. Despite significant advancement in face recognition technology, some areas require refinement for real-world applications. Specialized cameras can improve image quality and address filtering and restructuring

challenges. Face recognition technology has been widely utilized for access control purposes, security, and finance. However, it is now being introduced in areas such as logistics, retail, smartphones, transportation, education, real estate, and network information security. Face recognition has become a pivotal area of research within visual pattern recognition, leveraging neural networks to enable machines to process visual information as humans do. By analyzing matrices of pixels from photos or videos, face recognition systems identify individuals by detecting, positioning, and verifying facial features. These systems have broad applications, from access control and security to fields like logistics, retail, transportation, and network information security [1, 2].

Central to this study is the implementation of a face recognition system using an ESP32-CAM, incorporating Convolutional Neural Networks (CNNs) as the primary architecture. This device, paired with a TTL serial interface and an OLED display, facilitates real-time facial detection and recognition in resource-constrained environments. The inclusion of the OLED enhances data

visualization during operation, while the TTL serial interface ensures seamless communication with external devices.

The face recognition process typically involves face detection, feature extraction, and matching with a database. CNNs excel in such tasks, demonstrating high accuracy and robustness in extracting meaningful patterns from images [3]. Despite advancements, challenges such as varying lighting conditions, facial expressions, and partial obstructions persist in real-world applications [4]. Addressing these challenges necessitates innovative solutions, particularly in constrained hardware environments like the ESP32-CAM.

This paper explores the stages of developing and optimizing a face recognition system, from dataset preprocessing to model evaluation, emphasizing the effective integration of hardware components. By leveraging CNNs, the system aims to achieve high accuracy and real-time performance, making it suitable for diverse practical applications.

## 2. Related works

Saabia et al. [5] introduced an advanced face recognition framework that combines preprocessing, feature extraction, and selection processes, augmented by the application of the grey wolf optimization algorithm, and used the k-NN classifier. Muhammad Sajid et al. [6] explored the use of convolutional neural networks (CNN) for age-invariant face recognition using various architectures like AlexNet, VGGNet, GoogLeNet, and ResNet. Ab Wahab et al. [7] developed a hybrid CNN facial expression recognition model on the ESP-32 CAM, demonstrating the feasibility of implementing complex models on limited hardware. Additionally, Bajrami and Gashi [8] proposed a cost-effective and scalable solution for facial recognition using a CNN on ESP, addressing lighting conditions and image quality challenges. Gwyn et al. [9] used architectures such as VGG-16 and InceptionV3 to stand out for their high accuracy and F1 score, demonstrating the effectiveness of DL in improving facial recognition. Ariefwan et al. [10] made a comparison of CNN architectures, especially ResNet, MobileNetV2, and InceptionV3, for facial recognition systems. Dang and T. V. [11] introduced a new approach leveraging the ArcFace-based facial recognition model built into MobileNetV2. Moreover, Dang and T. V. [12] introduced an optimized model combining enhanced FaceNet with MobileNetV2 and SSD. By leveraging the strengths of different CNN architectures and integrating additional features, the field is moving towards more versatile, efficient, and accurate systems. The current research aims to contribute to this dynamic and, building on CNN models, break new ground in facial recognition technology using ESP32-CAM devices.

## 3. Methods

### 3.1. DL and Artificial Neural Network fundamentals

The neural network's fundamental architecture, comprising input, hidden, and output layers, is inspired by biological neurons in the human brain. Each artificial neuron connects to others in preceding and succeeding layers, without direct links within its layer. The number of hidden layers indicates the network's complexity and potential capabilities, as illustrated in Fig. 1. DNNs feature multiple hidden layers, enhancing complexity with interconnected units and neurons. Similar to brain neurons, neural networks evolve through learning, adjusting their data processing and analysis to improve efficiency and accuracy. Neurons receive multiple inputs and produce a single output, with connections represented by weight vectors indicating their strength. Inputs are multiplied by weights and adjusted by a bias parameter before passing through an activation function,

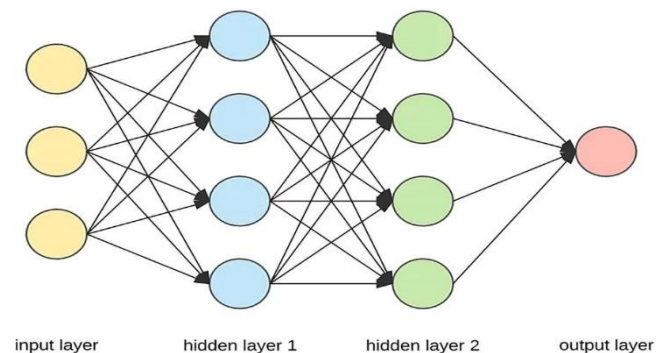


Figure 1. Artificial Neural Network with two hidden layers.

### 3.2. Principles of convolution neural networks

A CNN is a specialized kind of DNN, particularly adept at processing visual images. While DNNs generally have intricate architectures and necessitate vast datasets for training, they can comprise millions of parameters. This makes CNNs computationally more demanding than conventional methods. Despite their prowess in visual recognition tasks, CNNs might not always outperform traditional methods in terms of speed and resource efficiency. Figure 2 illustrates the fundamental architecture of a CNN, which encompasses key components: input, convolutional, pooling, dense, and output layers.

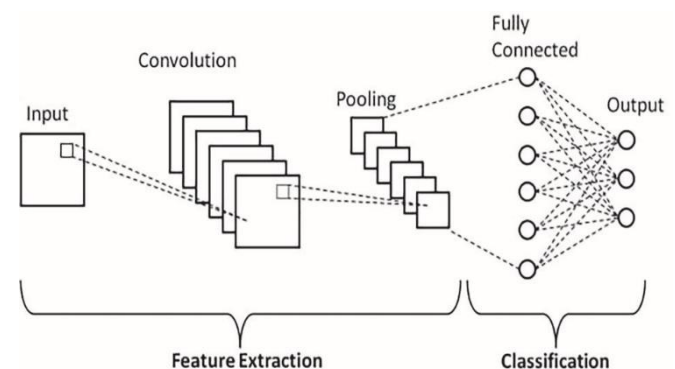


Figure 2. CNN diagram.

Table 1 provides a structured view of the different layers in a CNN, along with their respective functions and a description of what Figure

CNNs employ convolutional kernels, which are particularly adept at isolating local edge features. The primary motivation behind leveraging these convolutional mechanisms is to drastically minimize the number of parameters in play (in contrast to fully connected layers) while maintaining the proficiency to discern edge features on a local level. As a result, when pitted against DNNs, CNNs showcase enhanced efficiency during both the training and testing phases. A rudimentary convolutional kernel is illustrated in Fig. 3. For a CNN with a kernel size of  $2 \times 2$  and an input size of  $4 \times 3$ , with a stride of 1, the output dimensions are calculated as  $3 \times 2$ . If a fully connected layer were used for generating six output values (like the given example), it would entail 72 computations (i.e.  $4 \times 3 \times 3 \times 2$ ). In contrast, the CNN would require only 24 computations (i.e.  $3 \times 2 \times 2 \times 2$ ) to achieve the same.

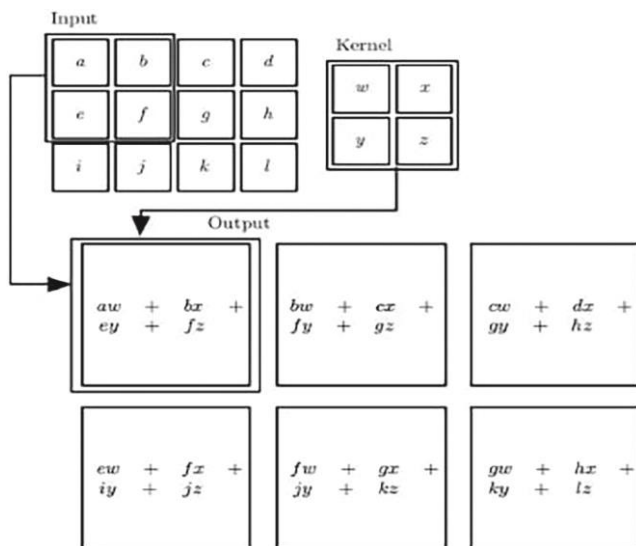


Figure 3. Convolutional kernel.

Here's the list of variables used in the operation in Fig. 3:

- $a-l$ : Elements of the input matrix. These could represent pixel intensities in an image.

- $w-z$ : Elements of the kernel matrix. These are the weights that will be learned during the training of a CNN.
- $aw + bx + ey + fz$ : represents the computation for the top-left value of the output matrix. They multiply the corresponding elements of the input matrix by the kernel weights and sum them up.

Element of the output matrix is calculated as follows:

- 1) For the top-left value in the output, you overlay the kernel on the top-left corner of the input matrix and compute:
- 2)  $(aw) + (bx) + (ey) + (fz)$
- 3) For the next value to the right in the output (first row, second column), you shift the kernel one step to the right and compute:
- 4)  $(bw) + (cx) + (dy) + (ez)$
- 5) This process continues across the entire input matrix to generate the output.

The output matrix is smaller than the input matrix because the kernel doesn't slide beyond the edges of the input matrix. If we want the output matrix to be the same size as the input, we would use padding around the input matrix.

CNNs undergo training by evaluating the accuracy of their predictions and making incremental adjustments to the model's parameters to refine these predictions. The model's performance is gauged using a loss function, which computes a 'loss' value based on the deviation between the predicted and actual outcomes. To enhance the model, this loss is then backpropagated through the network, guiding the adjustments to its parameters. This iterative process of forward and backward propagation continues until the model's performance plateaus or no longer shows significant improvement. To ensure the model generalizes well and doesn't overfit the training data, it's essential to assess its performance using a separate validation dataset, which represents unseen data by Scholkopf and Smola et al. [13].

Automatic face recognition has always been a subject of great intrigue. However, the multifaceted nature of face recognition makes it a challenging operation. In our investigation, we designed a CNN model, keeping certain parameters constant, such as filter size, pooling layers, and convolutional layers, but varying the architectural depth. This was to discern the effects of both depth and filter size on face recognition outcomes.

The size of the output can be derived based on several factors. Equation (1) to determine the weight of output O is presented as

$$O = \frac{W - K + 2P}{S} + 1 \quad (1)$$

Table 1. Layers in a CNN

Layer Type	Function	Description
Input Layer	Data entry	The first layer is where input image data is fed into the network.
Convolutional Layer	Feature extraction	Applies filters to the input to create a feature map, capturing local dependencies in the input.
Activation Layer	Non-linearity	Introduces non-linear properties to the system, allowing for complex patterns to be learned. Commonly uses ReLU (Rectified Linear Unit).
Pooling Layer	Dimensional reduction	Reduces the spatial size of the feature maps, thus reducing the number of parameters and computations in the network.
Fully Connected Layer	Classification	Neurons in this layer have full connections to all activations in the previous layer, as seen in regular Neural Networks, and are typically used for final classification.
Output Layer	Decision making	The final layer gives the output of the network, such as the class scores in classification tasks.

where:

- $W$  is the weight of the input.
- $K$  is the kernel size.
- $S$  is the stride.
- $P$  is the amount of zero-padding.
- $O$  is the weight of output.

The pooling layer functions by operating on each depth slice or channel of the input, modulating its spatial dimensions. In design, a max-pooling layer with a  $2 \times 2$  filter size  $F$  was employed, utilizing a stride  $S$  of 2. This procedure sub-samples every depth slice of the input by a factor of 2 in both its width and height dimensions, effectively discarding 75% of the activations. Equation (2) shows the relationship between the input and output weight sizes for such a pooling operation, which is encapsulated by

$$W_{\text{out}} = \frac{W_{\text{in}} - F}{S} + 1 \quad (2)$$

The Softmax function is used for multi-class classification problems and produces outputs between (0,1) that indicate the probability of each given input belonging to a class. Equation (3) below represents the Softmax function [14]:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } i = 1, \dots, k. \quad (3)$$

where:

- $\sigma$ : represents Softmax.
- $z$ : represents the input vector.
- $e^z$ : represents the standard exponential function of the input vector.
- $k$ : represents the number of classes in the multi-class classifier.
- $e^j$ : represents the standard exponential function of the output vector.

Figure 4. Block diagram of the face recognition system.

### 3.3. Face recognition system procedure

The details of the approach are presented in the following steps:

- 1) Step 1: initially, a subset of image samples from a labelled dataset is chosen and added to the training set, while the remaining samples are used to form a test set.
- 2) Step 2: the training set is then fed to the CNN model for training.
- 3) Step 3: the test set is subsequently recognized using the trained network from the above step.

In Fig. 4, the flowchart describes the facial recognition process using the Raspberry Pi 4. The process starts by loading the model, and then opening the webcam to take a photo. Images are preprocessed and analyzed for face detection. If no face is detected, it loops back to image capture. If a face is detected, it will extract and match features for facial recognition. After identification, it will display the results, then close the camera and end the process.

### 3.4. Transfer learning

Training a CNN network with such a massive dataset from scratch can be quite challenging. Instead of spending time, energy, and resources training all the layers of the model with random initial weights, it's more efficient to use a model pre-trained on a large dataset as a foundation for a new task.

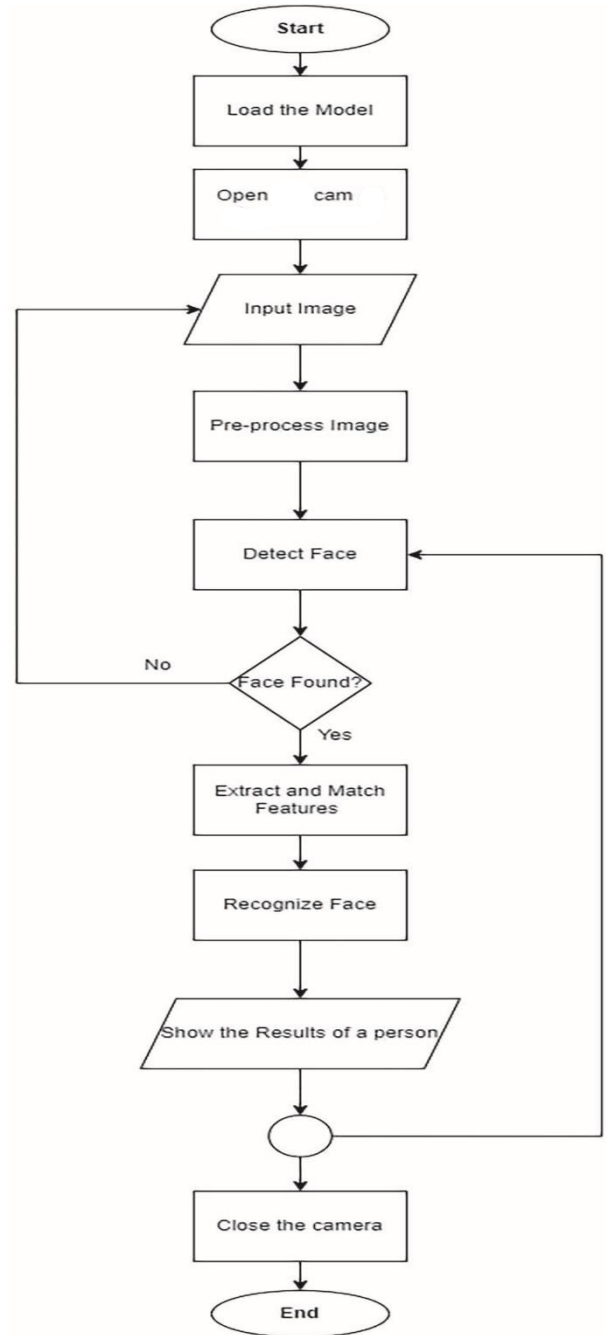


Figure 4. The flowchart of the Face recognition system

Moreover, training the system on a limited dataset can diminish the CNN model's ability to generalize. By transferring learned knowledge from one task in

one domain to another, achieving faster and more effective results is possible, a method termed 'transfer learning' by Pan and Yang [14]. The face recognition system utilizes the transfer learning method, leveraging a model with pre-existing learned features, to Figure 7. CNN models' parameters used in face recognition.

expedite the training process. Figure 6 illustrates the principle of transfer learning.

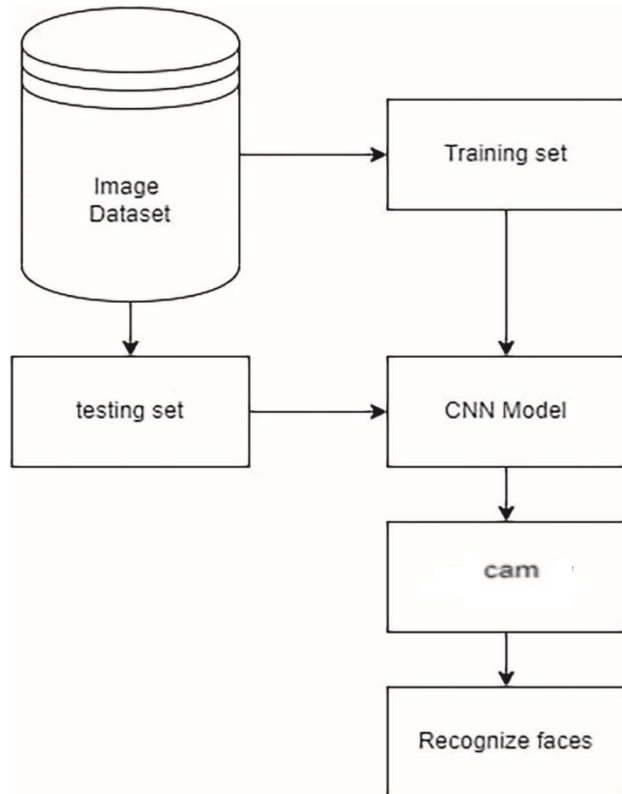


Figure 4. Block diagram of the face recognition system.

### 3.5. The proposed model

In this work, we utilized the principles of transfer learning to handle a face recognition problem by leveraging two pre-trained CNNs. Figure 7, the number of parameters used in the model architecture is presented.

In Table 2, the training models' setup is given. To reduce the time required for model convergence.

Figure 8 represents a face recognition pipeline. An input image goes through a base model comprising a pre-trained model that we use as feature extraction layers. The extracted features are then passed through a fully connected layer. The resulting features from the layer are fed into a softmax classifier for the final classification step, determining the identity of the person in the input image.

### 3.6. Dataset

We worked with both datasets (premade & custom):

The **premade** one consists of 2384 facial images that included celebrities' images. In this well-curated and distributed dataset, 1732 images were allocated for training and 652 for testing. The dataset divided into 12 distinct categories, with each category representing a different celebrity. We used the Celebrity Face Image Dataset from Kaggle, and the dataset is available from this [15] (The datasets were derived from the following source in the public domain: Vishesh Thakur (7 December 2022). *Celebrity Face Image Dataset, Version 1*. Retrieved 18 June 2024, from <https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset/data>),

The **custom** one consists of 846 facial images that included our faces dataset images. In this dataset, 618 images were allocated for training and 228 for testing. The dataset divided into 6 distinct categories, with each category representing a different person. Available from this [16], [https://drive.google.com/drive/folders/1MYoQKQ0fBL47YLzhTYphsmAnUM2syW\\_o](https://drive.google.com/drive/folders/1MYoQKQ0fBL47YLzhTYphsmAnUM2syW_o)

Table 6 describes the dataset features that are used in the face recognition system.

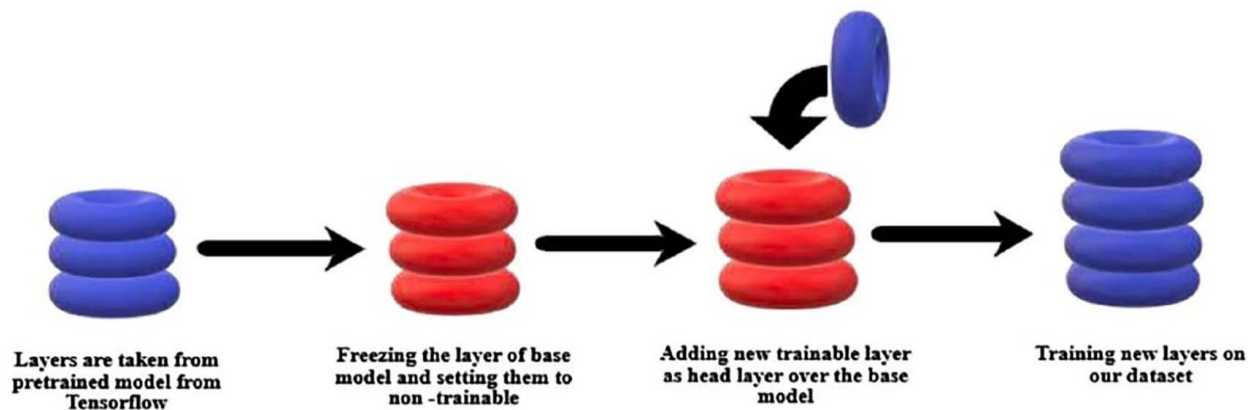


Figure 6. Principle of transfer learning.



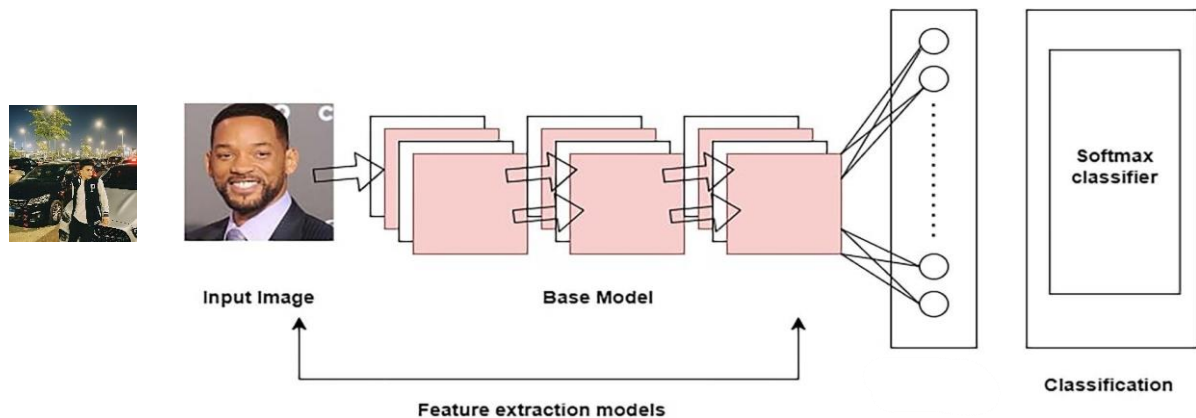
Figure 7. CNN architecture

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 253, 253, 32)	896
batch_normalization (BatchNormalization)	(None, 253, 253, 32)	128
max_pooling2d (MaxPooling2D)	(None, 126, 126, 32)	0
conv2d_1 (Conv2D)	(None, 124, 124, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 124, 124, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 60, 60, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295,168
batch_normalization_3 (BatchNormalization)	(None, 28, 28, 256)	1,024
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 1024)	51,381,248
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 6)	6,150

Table 2. Model training setup

Base Model	Top Layer	Batch Size	Epochs	Input Shape
CNN (Premade dataset)	Dense (128, ReLU)	32	100	128 × 128
CNN (Custom dataset)	Dense (1024, Relu)	64	50	256 × 256

Figure 8. Classification model training.



**Table 3.** Dataset features used in face recognition system

Attribute	Details
Total Size	2384 images for premade and 846 for custom
Image Resolution	128 × 128 pixels (premade) and 256 × 256 (custom)
Number of Categories	12 categories (celebrity) and 6 categories (us)
Images per Category	100 images per category (premade) and 50 (us)
Train Data Split	Training Set: 1732, 618 images (approximately 80%)
Test Data Split	Testing Set: 652, 228 images (approximately 20%)

**Table 4.** ESP-32 CAM features

Feature	Details
Launch Date	2019
Cores	Dual-core
Threads	2
Processor Frequency	240 MHz
Power Consumption	5V DC (average power usage is around 180–310 mA)
Memory	- 520 KB SRAM - 4 MB external PSRAM
Processor	Xtensa® 32-bit LX6 dual-core processor
Camera	OV2640 camera module (2MP resolution, supports JPEG image format)
Wireless Connectivity	Wi-Fi 802.11 b/g/n and Bluetooth 4.2 + BLE
GPIO Pins	9 programmable GPIO pins
Storage	Support for MicroSD card up to 4 GB
Interfaces	UART, SPI, I2C, PWM, ADC, DAC
Camera Interface	Support for an external camera module
Dimension	27 × 40.5 mm
Flash Memory	4 MB SPI Flash

### 3.8. ESP-32 CAM

The ESP32-CAM is a low-cost microcontroller-based module with a built-in camera that combines wireless communication capabilities with image capture. It is designed for IoT applications like facial recognition, video streaming, and remote monitoring. The module integrates an OV2640 2MP camera, which supports JPEG image format and provides decent image

quality for real-time applications. With its compact design and support for Wi-Fi and Bluetooth, the ESP32-CAM enables efficient image processing and transmission in resource-constrained environments.

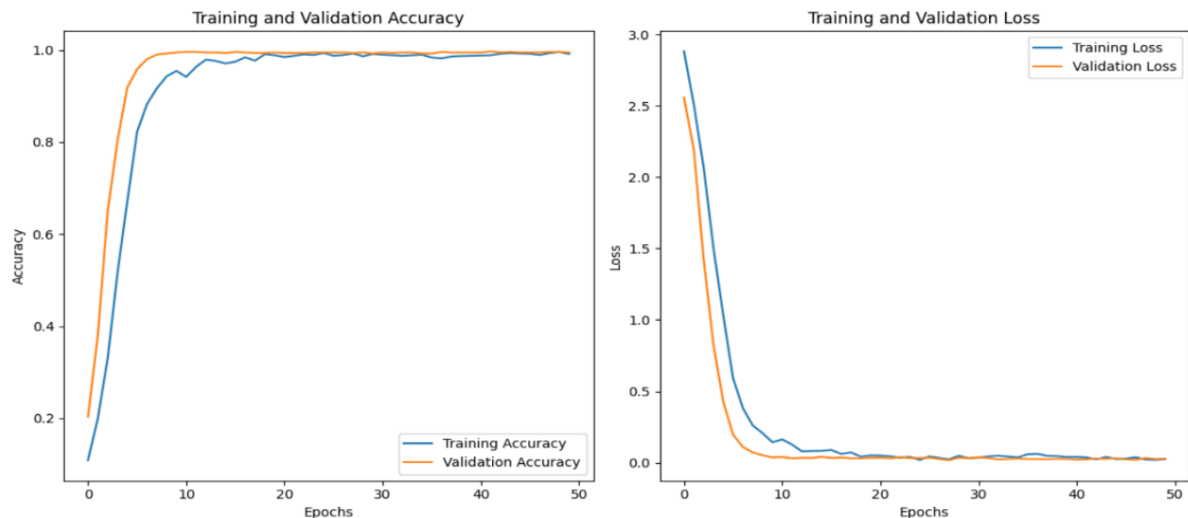
## 4. Experimental results

In this study, we utilized transfer learning techniques to train a model specifically for celebrity face recognition. We adopted the CNN models. We set up the ADAM optimizer with a technique to reduce the learning rate after each 10 epochs by 0.001. As shown in Fig. 9, we trained the CNN model for 50 epochs with premade dataset. Remarkably, after training, the model achieved a notable accuracy rate of 99.1%. This metric indicates that the model correctly identified the person in approximately 99.4% of the images in the test set. This high accuracy score is a testament to the model's capability to extract meaningful features from face images and correctly associate them with the corresponding face. The CNN model reported a loss value of 2.6% in the test set.

Figure 10 displays the accuracy of the CNN model for the custom dataset. After 50 epochs, the model achieved an accuracy of 94.6%. This means that the model correctly identified us in the test set with approximately 94.7% accuracy. This was slightly lower than the accuracy we achieved with the first model. However, this result indicates that the model effectively extracts significant features from face images and successfully matches these features with the respective labels. The second model recorded a training loss 2.6% in the test set.

Table 5 shows the accuracy and loss results of the models; After training the both datasets, First achieved a training accuracy of 99.1% and a training loss of 2.6%. Its validation accuracy is 99.4% and its validation loss is 2.6%. The model was trained for 50 epochs and took 186 minutes and 16.12 seconds. The second one achieved a training accuracy of 94.6% and a training loss of 2.6%. Its validation accuracy is 94.7% and its validation loss is 2.6%. It was also trained for 50 epochs but was slower, taking 290 minutes and 43.46 seconds.

Figure 11 shows the confusion matrix of the models. The CNN model's confusion matrix shows how often the model correctly predicts the class versus when it makes an error.

**Figure 9.** Accuracy and loss for premade dataset.

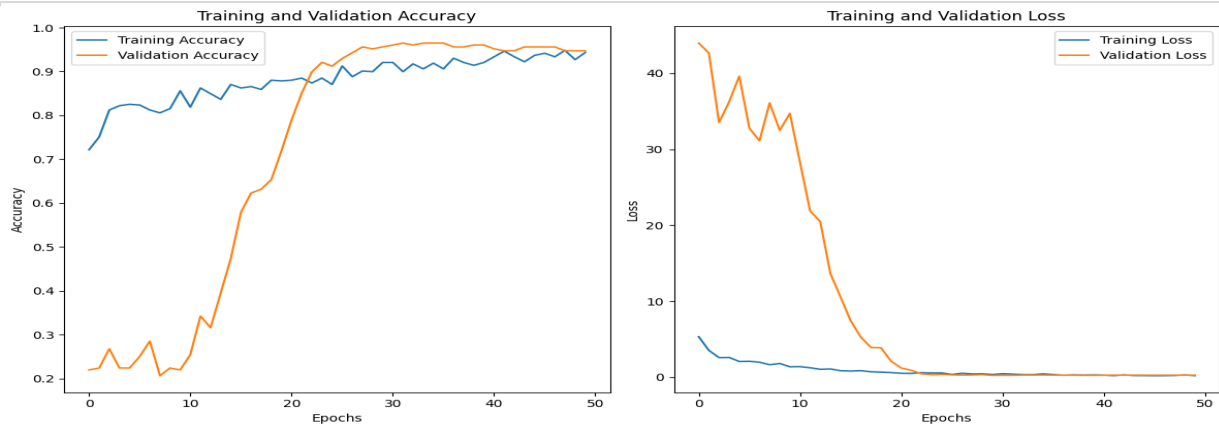


Figure 10. Accuracy and loss for custom dataset.

Table 5. Accuracy, loss, and time spent: results of the Models

Model	Training		Validation		Time spent
	Accuracy (%)	Loss (%)	Accuracy (%)	Loss (%)	
CNN (premade)	99.1	2.6	99.4	2.6	186 minutes and 16.12 seconds
CNN (custom)	94,6	2.6	94.7	2.6	290 minutes and 43.46 seconds

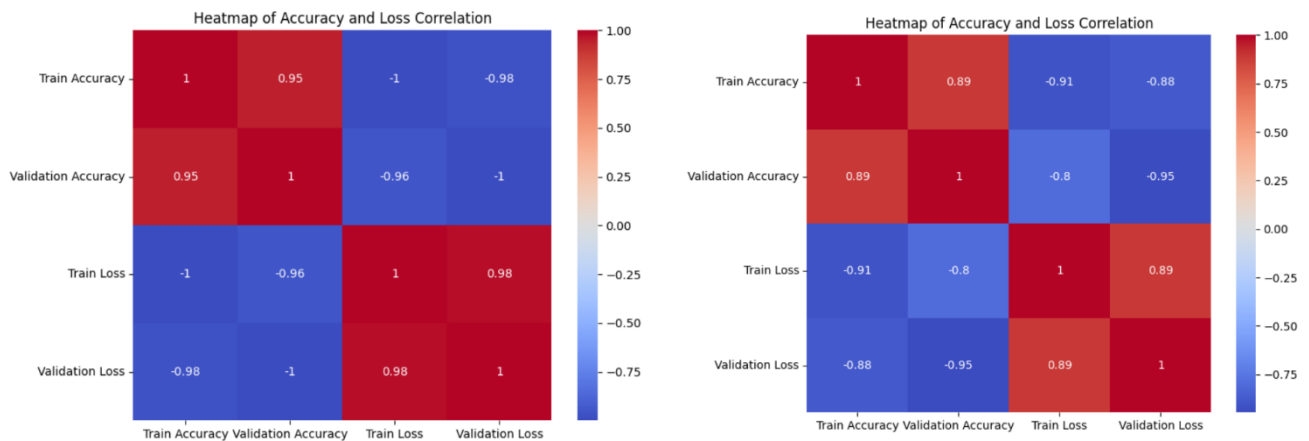


Figure 11. Confusion matrix of models.



Figure 12. Test results of face recognition system (premade dataset).





Figure 13. Test results of face recognition system.

Figure 12&13 display the test results, which show excellent precision in successfully classifying the faces of celebrities and our images.



Figure 14 display the used hardware components (ESP-32 cam, TTL and OLED).

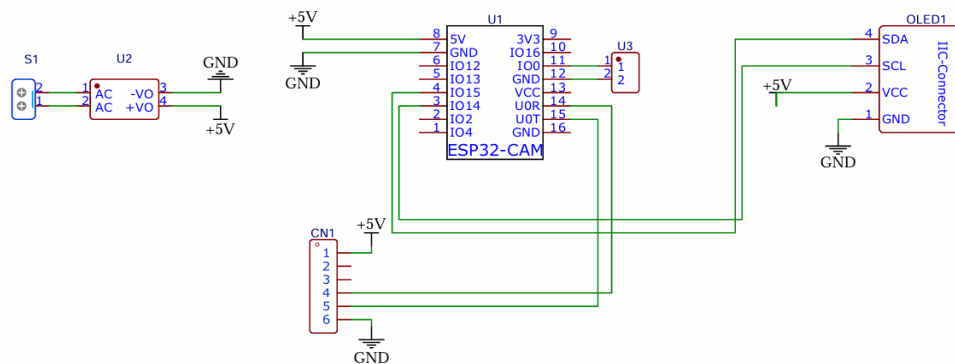


Figure 15 display the simulation of the ESP-32 cam.

## 5. Discussion

The study evaluated the performance of two CNN models, on the ESP-32 CAM. The model demonstrated high precision and recall, indicating better generalization to new data. One of the strengths of CNN is its high efficiency and excellent performance on lightweight devices like the ESP-32 CAM. However, limitations related to the ESP-32 CAM's computing power were noted. The duration of model training can be adjusted by adjusting the batch size. The study highlights CNN as the preferred choice for face recognition tasks due to its high efficiency and accuracy. These findings align with optimization efforts for lightweight devices, ensuring efficiency without sacrificing accuracy.

## 6. Conclusion

The project demonstrates the potential of using cost-effective hardware like ESP32-CAM for facial recognition tasks. By integrating neural networks, we successfully classified faces and laid the groundwork for practical applications. Moving forward, our vision of creating a drone-based smart attendance system highlights the innovative possibilities of combining AI and IoT with further development, this solution can transform how attendance is managed, making it more efficient, scalable.

## Ethics Statement

The images of celebrities in this study have been sourced from publicly available materials. We have provided proper attribution as referenced in the attached citation. No personal or private images were used, and our images were employed respectfully and ethically strictly for this study.

## Conflict of Interest Statement

The authors declare that there are no known financial or personal conflicts that could have influenced the results of this study.

## Data Availability Statement

The premade dataset were derived from the following source in the public domain: Vishesh Thakur (7 December 2022). Celebrity Face Image Dataset, Version 1. Retrieved 18 June 2024, from <https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset>.

The custom dataset: [https://drive.google.com/drive/folders/1MYoQKQ0fBL47YLzhTYphsmAnUM2syW\\_o](https://drive.google.com/drive/folders/1MYoQKQ0fBL47YLzhTYphsmAnUM2syW_o)

## FUTURE WORK

We will integrate an additional microcontroller like Raspberry Pi and we will apply techniques to enhance the performance and use improved models like VGG, Facenet and MobileNetV2.

## References

- [1] Y. Gurovich et al., "DeepGestalt - identifying rare genetic syndromes using deep learning," CoRR, abs/1801.07637.
- [2] V. S. Manjula and L. D. S. S. Baboo, "Face detection identification and tracking by PRDIT algorithm using image database for crime investigation," Int. J. Comput. Appl., vol. 38, no. 10, pp. 40–46, Jan. 2012.
- [3] K. Lander, V. Bruce, and M. Bindemann, "Use-inspired basic research on individual differences in face identification: Implications for criminal investigation and security," Cogn. Res., vol. 3, no. 1, pp. 1–13, Dec. 2018.
- [4] M. Wang and W. Deng, "Deep face recognition: A survey," Neurocomputing, vol. 429, pp. 215–244, 2021. <https://doi.org/10.1016/j.neucom.2020.10.081>. Lander K, Bruce V, Bindemann M. Use-inspired basic research on individual differences in face identification: implications for criminal investigation and security. *Cogn. Res.* 2018; 3:1–13. <https://doi.org/10.1186/s41235-018-0115-6>.
- [5] Saabia AA-B, El-Hafeez T, Zaki AM. Face recognition based on Grey wolf optimization for feature selection. Proc. Int. Conf. Adv. Intell. Syst. Inform. (AISI) 2018; 845:273–283. [https://doi.org/10.1007/978-3-319-99010-1\\_25](https://doi.org/10.1007/978-3-319-99010-1_25).
- [6] Sajid M, Ali N, Ratyal NI. et al. Deep learning in age-invariant face recognition: a comparative study. Comput. J. 2022; 65:940–972. <https://doi.org/10.1093/comjnl/bxaa134>.
- [7] Ab Wahab MN, Nazir A, Ren ATZ. et al. Efficientnet-lite and hybrid CNN-KNN implementation for facial expression recognition on Raspberry Pi. IEEE Access 2021; 9:134065–134080. <https://doi.org/10.1109/ACCESS.2021.3113337>.
- [8] Bajrami and Gashi, "A cost-effective and scalable solution for facial recognition using a DNN, addressing lighting conditions and image quality challenges."
- [9] Gwyn T, Roy K, Atay M. Face recognition using popular deep net architectures: a brief comparative study. Future Internet 2021; 13:164. <https://doi.org/10.3390/fi13070164>.
- [10] Ariefwan MRM, Diyasa IGSM, Hindrayani KM. CNN performance comparison on face recognition classification. Literasi Nusantara 2021; 4: 1–10.
- [11] Dang T-V. Smart home management system with face recognition based on ArcFace model in deep convolutional neural network. J. Robot. Control 2022; 3:754–761. <https://doi.org/10.18196/jrc.v3i6.15978>.
- [12] Dang T-V. Smart attendance system based on improved facial recognition. J. Robot. Control 2023; 4:46–53. <https://doi.org/10.18196/jrc.v4i1.16808>.

- [13] Gwyn T, Roy K, Atay M. Face recognition using popular deep net architectures: a brief comparative study.*Future Internet* 2021; **13**:164.  
<https://doi.org/10.3390/fi13070164>.
- [14] Ali AA, El-Hafeez T, Mohany Y. A robust and efficient system to detect human faces based on facial features.*Asian J.Res.Comput. Sci.* 2019; **2**:1–12.  
<https://doi.org/10.9734/ajrcos/2018/v2i430080>.
- [15] Thakur V.Celebrity face image dataset,version 1.Retrieved June 18, 2024 from <https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset/data>.
- [16] [https://drive.google.com/drive/folders/1MYoQKQ0fBL47YLzhTYphsmAnUM2syW\\_o](https://drive.google.com/drive/folders/1MYoQKQ0fBL47YLzhTYphsmAnUM2syW_o)