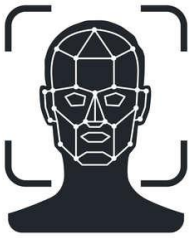
Facial Recognition project

Arya Krishna



Project Description

This project presents a real-time face recognition and access control system that uses artificial intelligence to identify individuals based on facial images. By training a computer model to recognise patterns in human faces, the system can determine whether someone is known and decide whether to grant access.

The system works by comparing a captured face—either from a webcam or image file—with a collection of previously seen faces. If a match is found with high confidence, access is granted; otherwise, it is denied. The project also includes visual demonstrations showing how the model distinguishes between matching and non-matching faces.

This work showcases how modern technologies such as machine learning, computer vision, and deep learning can be combined to solve practical problems. It is designed to be a stepping stone for further enhancements such as emotion detection, improved security systems, and real-world deployments.

Table of Contents

2. Data Preprocessing and Sample Output

An overview of the data cleaning, organisation, and augmentation process, followed by example outputs demonstrating model predictions.

3. Modelling and Training Approach

A description of the model architecture, training methodology, use of transfer learning, and discussion of observed limitations.

4. Conclusion

A summary of key outcomes, reflections on system performance, and recommendations for future improvements.

How to Use the Face Recognition and Access Control System

This section outlines the steps required to install, configure, and use the real-time face recognition and access control system. The system is designed to recognise users based on facial images and grant or deny access depending on whether the individual is present in the trained dataset.

1. Preparing the Dataset

To begin using the system, the user must create a dataset of facial images. This is done by creating a folder named dataset within the project directory. Inside this folder, separate subfolders should be created for each individual the system should recognise. Each subfolder must be named after the person (for example, Aryak) and contain between five to ten images of that person's face.

Images should be of good quality and include slight variations in angle, lighting, or facial expression to allow the model to generalise effectively. All images must be saved in .jpg format and consistently named (for instance, Aryak_0001.jpg, Aryak_0002.jpg, etc.).

2. Installing the Required Packages

Before the system can be used, all dependencies must be installed. These include libraries such as TensorFlow, OpenCV, DeepFace, NumPy, Pandas, and Matplotlib. These packages can be installed using the Python package manager by running the following command in the terminal or command prompt:

bash
CopyEdit
pip install -r requirements.txt

This command installs all necessary dependencies listed in the requirements.txt file that comes with the project.

3. Training the Face Recognition Model

After preparing the dataset, the next step is to train the facial recognition model. This is done by executing the model_training.py script. The training process involves loading the image data, applying augmentation techniques to artificially expand the dataset, and training a convolutional neural network (CNN) or a pretrained MobileNetV2 model to classify faces.

The script also saves the trained model for later use and generates plots to visualise training and validation performance. These plots can help diagnose overfitting or underfitting and guide further tuning of the model.

4. Running the Access Control System

Once the model has been trained, the access control system can be launched by running the access_control.py script. When executed, this script opens the system's webcam and displays a live feed. The user can press the space bar to capture an image, which is then passed through the trained model.

If the model successfully identifies the individual and their confidence score exceeds a predefined threshold, the system prints a message indicating that access has been granted. If the individual is not recognised or the confidence is too low, the system denies access.

5. Optional: Running Emotion Detection

In addition to facial recognition, the system can also detect a user's emotional state. This is achieved using the DeepFace library, which leverages pretrained models to classify

emotions from facial expressions. The emotion detection system is started by running the emotion_detector.py script.

Once launched, the webcam feed opens, and the user can press the space bar to analyse their current emotional expression. The script outputs the most likely emotion, such as happy, sad, angry, or neutral. This feature is optional and can be used to enhance the access system or provide additional contextual information.

6. Additional Features and Extensions

This project can be extended in a number of directions. For example, voice feedback can be added using a text-to-speech library, allowing the system to audibly announce access decisions. The system can also be integrated with physical hardware such as an electric door lock or Raspberry Pi GPIO pins to create a complete entry system.

Another possible extension is to log access attempts by saving captured images, prediction results, and timestamps. Additionally, the facial recognition component could be replaced or complemented with an embedding-based model such as FaceNet, which is capable of recognising faces even with very limited training images.

Summary

In summary, the system allows users to prepare a dataset of known individuals, train a face recognition model, and use it for real-time access control through a webcam. With optional emotion detection and opportunities for hardware integration, this project serves as a practical foundation for a range of facial analysis applications.

Data Preprocessing and Sample Output

To enable accurate face recognition, the dataset was first organised into subdirectories, with each folder corresponding to a specific person. Each image was resised to a uniform resolution of 100×100 pixels to maintain consistency across the dataset. A custom script was used to parse the dataset and extract labeled pairs of images for both identification and verification tasks. The images were also normalised by scaling pixel values to the [0, 1] range. For the verification task in particular, additional preprocessing steps were taken to create structured dataframes that grouped image pairs and assigned binary labels indicating whether the two images depicted the same person or not.

A significant challenge during preprocessing was the inconsistency in data formatting and image naming conventions. Many images were stored deep within nested folders, and some expected files were missing or misnamed, leading to initial failures in path construction and image loading. Furthermore, class imbalance and the limited number of training images for certain individuals posed risks of overfitting and poor generalization. These limitations were partially addressed through data filtering (removing classes with fewer than two images) and augmentation techniques.

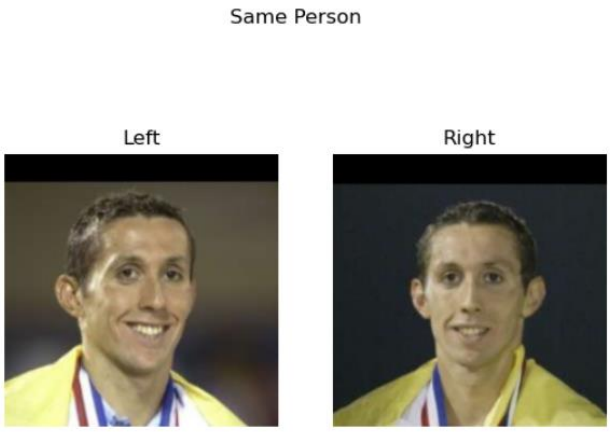


Figure 1: Same Person

The figure 1 above presents an example of a positive match generated by the system. The left and right images show the same individual under slightly different lighting conditions and angles. The model correctly classified this pair as "Same Person," validating the effectiveness of the preprocessing and learning pipeline. However, the system's performance varied depending on the quality and diversity of the training data, highlighting the importance of balanced and representative input during model training.

Figure 2: Different People

Figure 2 above presents an example in which the system correctly identifies two different individuals. The images on the left and right depict distinct people with no shared identity, and the model has accurately classified this pair as “Different People.” This result demonstrates the model's capacity to distinguish between dissimilar facial features, even when presented with varying expressions, backgrounds, or lighting conditions. The ability to correctly reject mismatched pairs is just as important as recognising matching ones, especially in real-world access control scenarios. Although visual differences in this case are clear to the human eye, the model must rely solely on learned feature representations to make such distinctions. This example underscores the effectiveness of the model’s decision boundaries and its ability to generalise to unseen combinations during inference.

Modelling and Training Approach

The modelling component of the system was initially implemented using a custom convolutional neural network (CNN) designed to classify faces into predefined identity categories. The architecture consisted of multiple convolutional and max pooling layers, followed by a flattening layer and a final dense layer with softmax activation to predict the most probable identity. However, due to the limited size and variability of the dataset, this baseline model began too overfit quickly. This was evident in the training metrics, which showed a steep rise in training accuracy alongside a persistently low validation accuracy.

To mitigate overfitting and encourage better generalisation, data augmentation techniques were introduced. These included transformations such as random rotation, width and height shifts, zooming, and horizontal flipping. The goal was to artificially expand the dataset and expose the model to a wider variety of input conditions. Although data augmentation did provide marginal improvements, it was insufficient on its own, especially when the model was trained on many individuals with only a few samples each.

To address these limitations more effectively, a transfer learning approach was adopted using MobileNetV2 as a pretrained feature extractor. The base model was frozen to preserve its learned weights, and a custom classification head was added for identity prediction. This significantly improved the model’s ability to generalise, as it could leverage low-level facial features learned from a large-scale dataset such as ImageNet. The model was trained using the Adam optimiser and categorical cross-entropy loss, with validation monitored during training to avoid unnecessary overtraining.

Despite these enhancements, the final training metrics revealed ongoing challenges with overfitting as can be seen in figure 3. As illustrated in the accompanying accuracy and loss

plots, the model achieved near-perfect performance on the training set, but validation accuracy remained low and validation loss steadily increased. This divergence suggests that even with transfer learning and data augmentation, the model struggled to generalise due to the small sample size and imbalance in class representation. For future improvement, embedding-based face recognition methods such as FaceNet or ArcFace could be explored, as they are more robust to class imbalance and require fewer images per identity

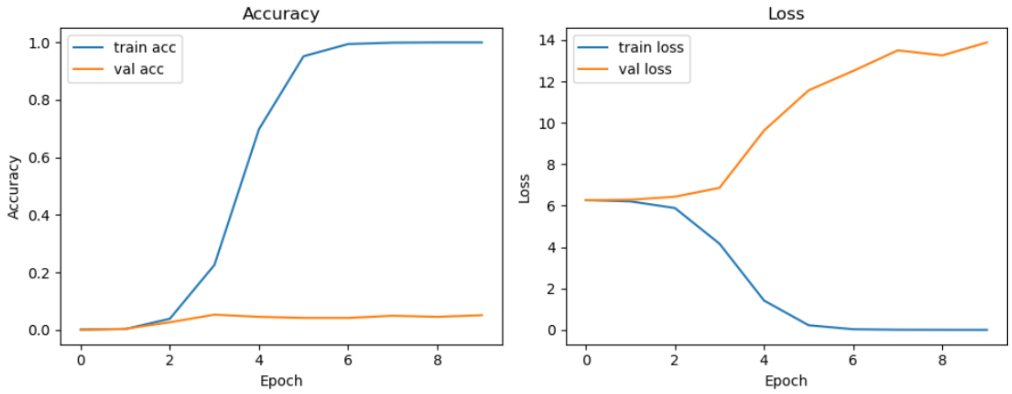


Figure 3: Train and validation accuracy and loss curves

Conclusion

This project successfully demonstrated the development of a real-time face recognition and access control system using deep learning techniques. Through the use of convolutional neural networks and transfer learning with MobileNetV2, the system was able to identify individuals based on facial images and determine whether access should be granted. A structured data preprocessing pipeline, combined with data augmentation, enabled the model to learn meaningful facial representations despite a relatively small dataset.

While the model achieved high training accuracy, the validation results revealed limitations in generalisation, particularly due to class imbalance and the limited number of samples per identity. This was most evident in the divergence between training and validation performance. Nevertheless, the system was functional in practice and capable of correctly distinguishing between matching and non-matching face pairs, as evidenced by the output examples provided.

Additional functionality, such as emotion detection using a pre-trained DeepFace model, further extended the application's potential use cases. Future improvements may include adopting embedding-based recognition methods such as FaceNet, expanding the dataset, and integrating with physical access hardware for deployment in real-world environments.

Overall, the project illustrates the viability of combining machine learning, computer vision, and real-time systems to create practical and extensible facial recognition applications.