Πανεπιστήμιο Πειραιώς
ΕΚΕΦΕ "ΔΗΜΟΚΡΙΤΟΣ"
Δ.Π.Μ.Σ. στην Τεχνητή Νοημοσύνη
Τμήμα Ψηφιακών Συστημάτων Πανεπιστημίου Πειραιώς
Ινστιτούτο Πληροφορικής και Τηλεπικοινωνιών ΕΚΕΦΕ «Δημόκριτος»

Εργασία στο μάθημα

"Βαθιά Μηχανική Μάθηση"

Facial recognition & classification using CNNs

Βιτζηλαίος Βασίλειος

MTN2201

Athens, 2023

Table of Contents

Abstract

This project presents the development of a real-time face recognition system using deep learning techniques. The project involves dataset preparation, model training, and real-time face recognition, aiming to create an accurate and efficient solution. Three distinct models are implemented and evaluated using metrics such as accuracy, precision, recall, and F1-score. The trained models exhibit adequate performance in detecting and recognizing faces in real-time video streams. The project contributes to the field of face recognition by harnessing the potential of deep learning algorithms and providing a practical solution for real-time applications. The system holds promise for integration into security systems, surveillance networks, and personalized user experiences, showcasing its wide-ranging implications in various domains.

Introduction

Face recognition is a critical technology that has gained significant attention and applications in various domains, including security systems, surveillance, biometrics, and human-computer interaction. The ability to automatically detect and recognize faces in real-time scenarios holds immense value in enhancing security, improving user experiences, and enabling personalized services. This project focuses on developing an accurate and efficient face recognition system using deep learning techniques.

The primary objective of this project is to build a robust face recognition model capable of accurately detecting and recognizing faces in real-time. The project encompasses several key components, including dataset preparation, model architecture design, model training, real-time face recognition, and metrics visualization. By leveraging deep learning algorithms, the project aims to achieve high accuracy and real-time performance, enabling seamless integration into various applications and systems.

The dataset used for training and evaluation plays a crucial role in the success of face recognition systems. This project utilizes a dataset consisting of images captured by the camera's live feed. When the application starts, a window opens, projecting what camera is seeing with the closest face being framed. The calculation is based on the size of the face as it is not possible for a single RGB camera to determine depth of view. The user has the choice to take a shot by pressing down the 'S' button of their keyboard. To exit this process, the user can hit the 'Q' button.

The project encompasses multiple model architectures, each designed to capture and extract discriminative features from face images. These models consist of convolutional layers for feature extraction and fully connected layers for classification. Activation functions such as ReLU introduce non-linearity, and pooling operations enable down-sampling and abstraction of features. The models' designs are aimed at achieving a balance between model complexity, computational efficiency, and performance.

Model training is a critical step in building an effective face recognition system. During training, the models' parameters are optimized using mini-batch gradient descent and the cross-entropy loss function. The training process involves iterative updates to the model's weights based on the gradients computed during backpropagation. To prevent overfitting and ensure generalization, validation is performed after each epoch. We also incorporated an early stoppage mechanism in addition to the above functionality. The trained models' weights are saved for future use and deployment.

Real-time face recognition is a key aspect of the project, enabling the system to operate in real-time scenarios. The trained models are applied to frames captured from a video stream, and face detection is performed using OpenCV's Haar cascades. Detected faces are preprocessed, passed through the models, and classified into appropriate classes. The predicted labels are overlaid on the frames, providing real-time face recognition capabilities for applications such as access control, identity verification, and personalized experiences.

Metrics visualization plays an essential role in evaluating and understanding the performance of the face recognition system. The project provides functionality to plot various metrics, including training and validation loss, validation accuracy, precision, recall, F1-score, and confusion matrices. These visualizations aid in analyzing the models' behavior, identifying trends, and assessing their efficacy in different scenarios.

Methodology

The methodology employed for the development of the real-time face recognition system using deep learning techniques can be outlined as follows.

1. Dataset Preparation

The project begins with the collection of a diverse dataset comprising face images representing various individuals, facial expressions, poses, and lighting conditions. The dataset is then split into training and validation subsets, ensuring an appropriate distribution of samples for model training and evaluation purposes. Data augmentation techniques, such as resizing, random horizontal flipping, rotation, and affine transformations, are applied to augment the dataset and enhance the model's ability to generalize.

2. Model Architecture Design

Three different model architectures are designed specifically for face recognition. These models incorporate convolutional layers for feature extraction and fully connected layers for classification. Activation functions and pooling layers are carefully selected and integrated into the models to facilitate effective feature extraction.

Model 1

Model 2

Input: The model takes as input images with three color channels (RGB).

Convolutional Layers:

Layer	Input size	Filter size	Kernel size
Conv1	3	64	3
Conv2	64	128	3
Conv3	128	256	3

Convolution Layer 1

- **Conv1**: Applies a 2D convolution with 64 filters, each having a kernel size of 3x3. This layer extracts low-level features from the input images.
- **ReLU1**: Applies the Rectified Linear Unit (ReLU) activation function to introduce non-linearity to the network.
- **MaxPool1**: Performs 2D max pooling with a kernel size of 2x2 to down-sample the spatial dimensions of the feature maps by half. This reduces the model's sensitivity to the precise spatial location of features.

Convolution Layer 2

- **Conv2**: Applies a 2D convolution with 128 filters and a kernel size of 3x3. This layer further extracts more complex features from the input.
- **ReLU2**: Applies the ReLU activation function.
- **MaxPool2**: Performs 2D max pooling with a kernel size of 2x2.

Convolution Layer 3

- **Conv3**: Applies a 2D convolution with 256 filters and a kernel size of 3x3. This layer continues to extract higher-level features.
- **ReLU3**: Applies the ReLU activation function.
- **MaxPool3**: Performs 2D max pooling with a kernel size of 2x2.

Flatten: The feature maps are flattened into a 1D vector to be fed into the fully connected layers.

Fully Connected Layers:

Layer	Input size	Output size
FC1	256*30*30	120
FC2	120	84
FC3	84	Num_classes

Fully Connected Layer 1

- **FC1**: Applies a linear transformation to the flattened input with 230400 input features and 120 output features.
- **ReLU4**: Applies the ReLU activation function.

Fully Connected Layer 2

- **FC2**: Applies a linear transformation with 120 input features and 84 output features.
- **ReLU5**: Applies the ReLU activation function.

Output Layer

- FC3 applies a linear transformation to produce the final output logits with several units equal to the number of classes. The model predicts the class probabilities using these logits.

Model 3

Input: The model takes as input images with three color channels (RGB).

Convolutional Layers:

Layer	Input size	Filter size	Kernel size
Conv1	3	64	3
Conv2	64	128	3
Conv3	128	256	3

Convolution Layer 1

- **Conv1**: Applies a 2D convolution with 64 filters, each having a kernel size of 3x3. This layer extracts low-level features from the input images.
- **BatchNorm1**: Performs batch normalization on the output of Conv1, which normalizes the activations to help with training stability and faster convergence.
- **ReLU1**: Applies the Rectified Linear Unit (ReLU) activation function to introduce non-linearity to the network.
- **MaxPool1**: Performs 2D max pooling with a kernel size of 2x2 to down-sample the spatial dimensions of the feature maps by half. This reduces the model's sensitivity to the precise spatial location of features.

Convolution Layer 2

- **Conv2**: Applies a 2D convolution with 128 filters and a kernel size of 3x3. This layer further extracts more complex features from the input.
- **BatchNorm2**: Performs batch normalization on the output of Conv2.
- **ReLU2**: Applies the ReLU activation function.
- **MaxPool2**: Performs 2D max pooling with a kernel size of 2x2.

Convolution Layer 3

- **Conv3**: Applies a 2D convolution with 256 filters and a kernel size of 3x3. This layer continues to extract higher-level features.
- **BatchNorm3**: Performs batch normalization on the output of Conv3.
- **ReLU3**: Applies the ReLU activation function.
- **MaxPool3**: Performs 2D max pooling with a kernel size of 2x2.

Flatten: The feature maps are flattened into a 1D vector to be fed into the fully connected layers.

Fully Connected Layers:

Layer	Input size	Output size
FC1	256*30*30	120
FC2	120	84
FC3	84	Num_classes

Dropout: Applies dropout regularization with a probability of 0.5. Dropout randomly sets a fraction of input units to 0 at each training step, which helps prevent overfitting.

Fully Connected Layer 1

- **FC1**: Applies a linear transformation to the flattened input with 230400 input features and 120 output features.
- **BatchNorm4**: Performs batch normalization on the output of FC1.
- **ReLU4**: Applies the ReLU activation function.

Fully Connected Layer 2

- **FC2**: Applies a linear transformation with 120 input features and 84 output features.
- **BatchNorm5**: Performs batch normalization on the output of FC2.
- **ReLU5**: Applies the ReLU activation function.

Output Layer

- FC3 applies a linear transformation to produce the final output logits with several units equal to the number of classes. The model predicts the class probabilities using these logits.

3. Model Training

The models' parameters are initialized, and an optimization algorithm, such as Stochastic Gradient Descent with momentum, is employed. A suitable for multiclass image classification loss function, typically cross-entropy, is defined to quantify the discrepancy between predicted and actual class labels. The models are trained using mini-batch gradient descent on the augmented training dataset. The learning rate and the number of training epochs are adjusted based on performance analysis and convergence criteria.

Loss function	Cross-Entropy
Optimization algorithm	Stochastic Gradient Descent
• **Learning rate**	0.001
• **Momentum**	0.9

4. Performance Evaluation

The trained models are evaluated on the validation dataset to assess their classification performance. Evaluation metrics such as accuracy, precision, recall and F1-score are calculated to measure the models' effectiveness. Furthermore, confusion matrices are generated to gain insights into the models' performance across different face classes. The performance metrics and confusion matrices are visualized to facilitate a comprehensive analysis of the models' behavior and performance trends.

Training and Validation Scores

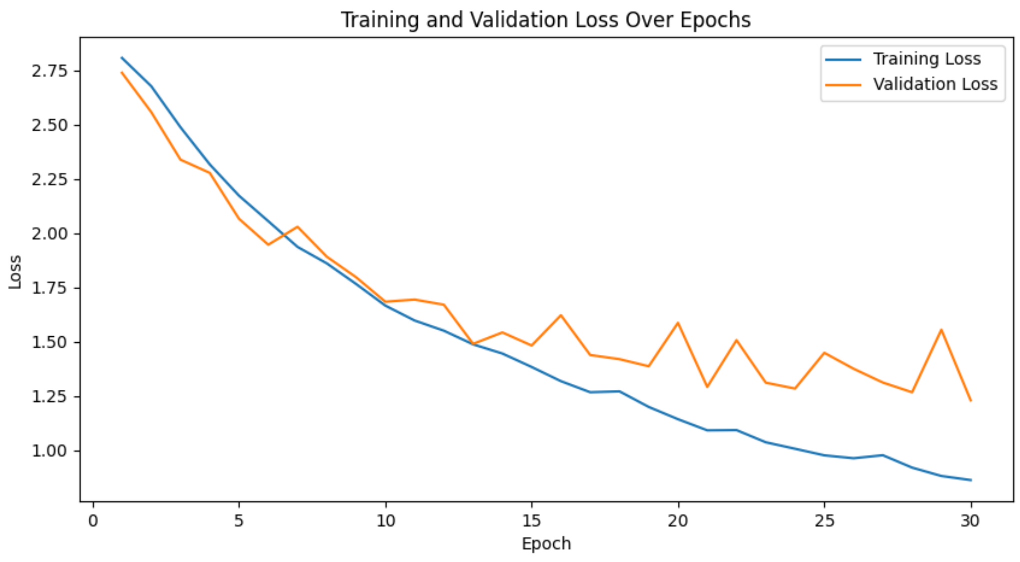


Figure 1 - model 1 training and validation loss

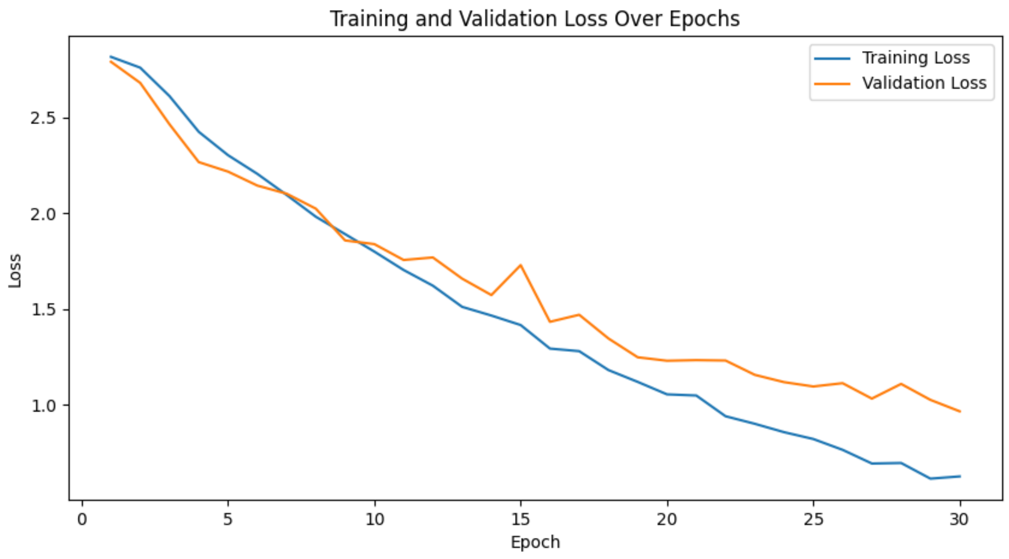


Figure 2 - Model 2 training and validation loss

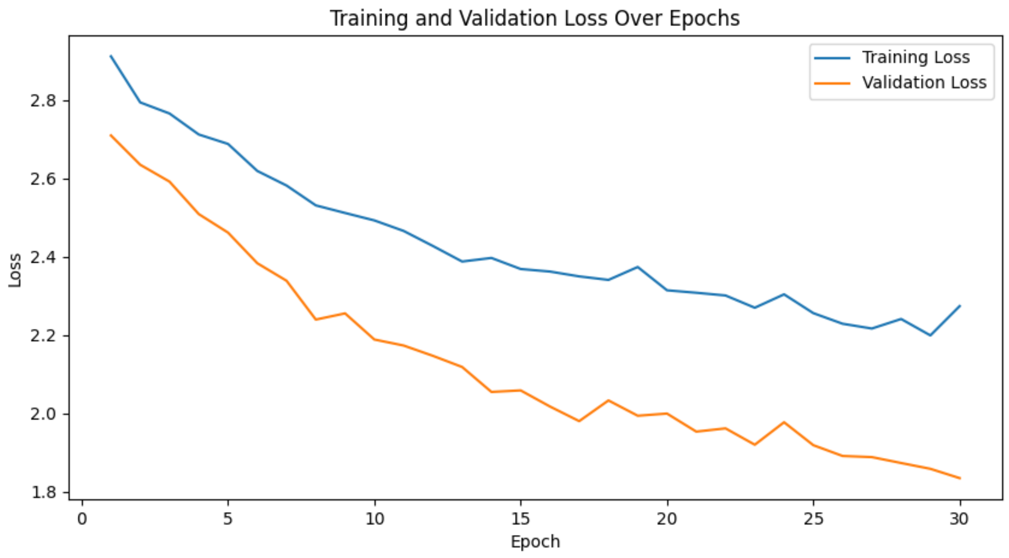


Figure 3 - Model 3 training and validation loss

Model 2 demonstrates better performance compared to models 1 and 3, both in terms of convergence and avoiding overfitting. Specifically, model 1 starts to exhibit signs of overfitting after approximately 15 - 20 epochs, while model 2 maintains a more balanced and stable convergence throughout the training process.

Precision, Accuracy, Recall and F1 scores

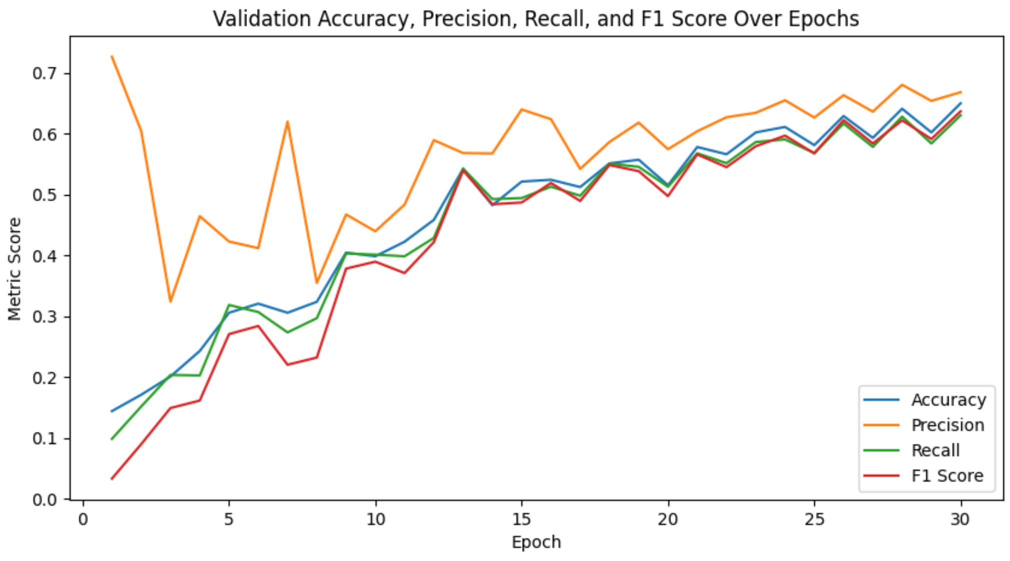


Figure 4 - Model 1 metric scores

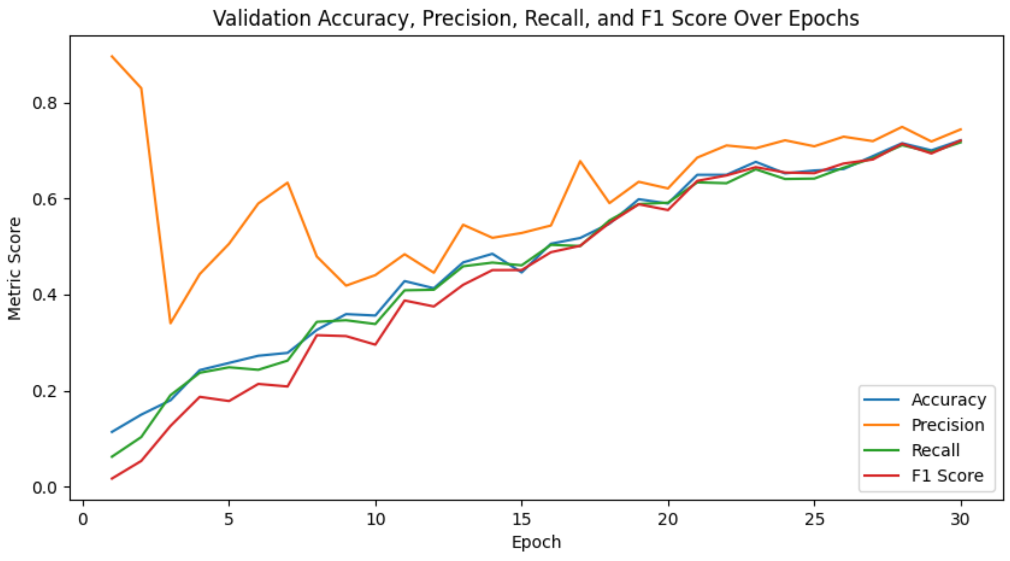


Figure 5 - Model 2 metric scores

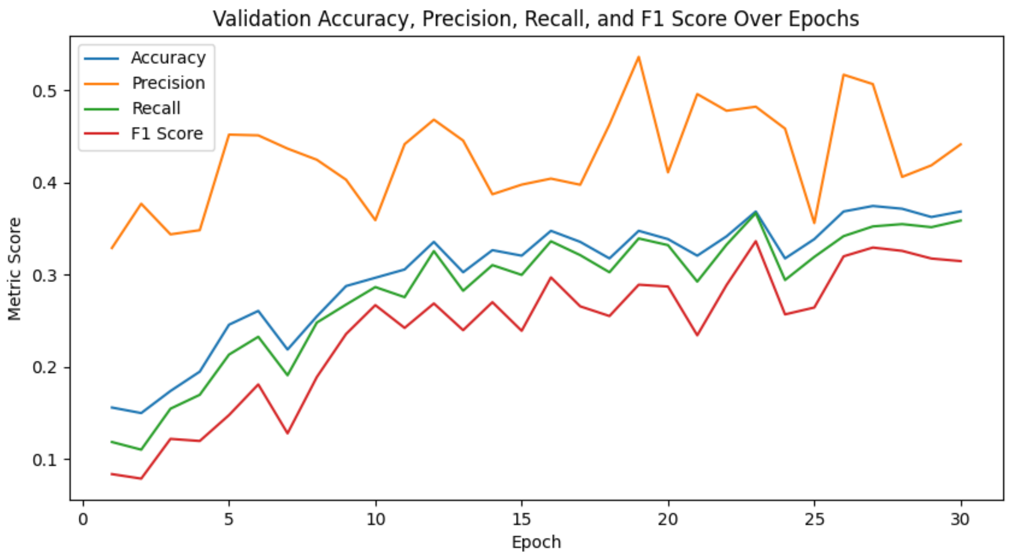


Figure 6 - Model 3 metric scores

The experimental results reveal an interesting observation regarding the relationship between model complexity and performance. Contrary to expectations, the more complex Model 3 did not consistently outperform the other models. In fact, its performance scores exhibited higher variability throughout the training process compared to the relatively simpler Model 1 and Model 2.

These findings highlight the importance of not solely relying on model complexity as a determinant of performance. The stability and convergence of the models, as observed through the training and validation losses, can provide valuable insights into their overall effectiveness. Thus, a more comprehensive analysis considering both accuracy and training dynamics can guide the selection and refinement of face recognition models.

Confusion Matrices

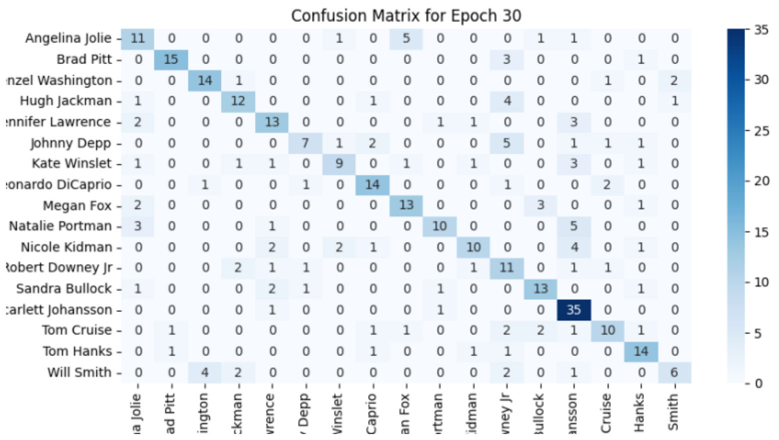


Figure 7 - Model 1 confusion matrix

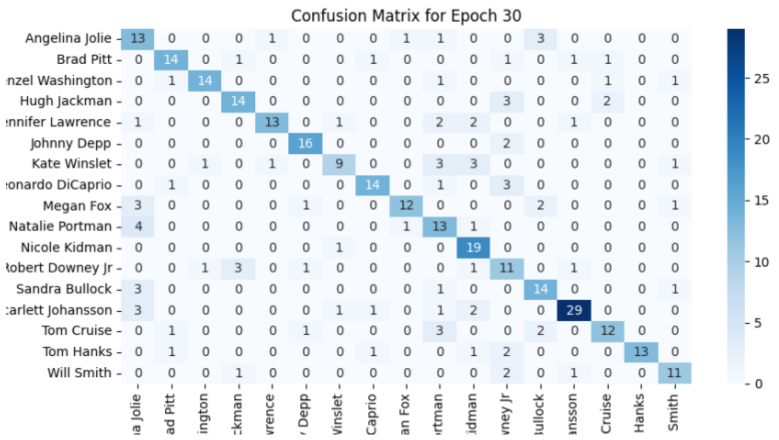


Figure 8 - Model 2 confusion matrix

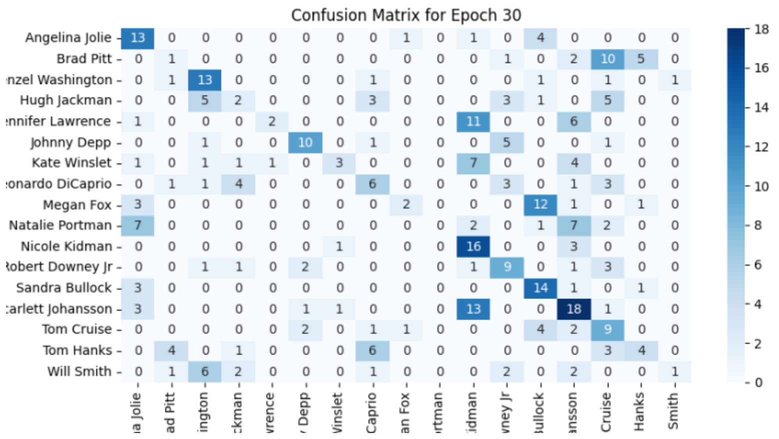


Figure 9 - Model 3 confusion matrix

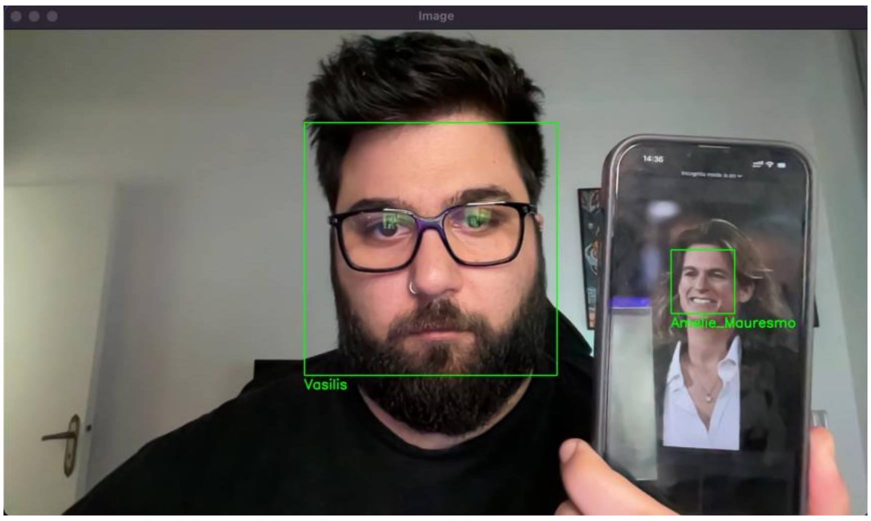
Upon examining the confusion matrices, our initial suspicions regarding the performance of model 2 are confirmed. The matrices reveal a more distinct diagonal pattern among the classes, indicating improved accuracy and stability in the results. In contrast, while model 1 also demonstrates promising results, the potential for overfitting observed during the training phase raises concerns and leads us to exclude it from our final selection of the most suitable model. As for model 3, its performance remains significantly below our desired estimation, further highlighting its limitations.

5. Model Selection

Now that the models got compared based on the evaluation metrics, we have a clearer view of what is going on. The model demonstrating the highest accuracy and desired performance trade-offs is selected for further deployment and integration into real-world applications. With that said and based on the above metrics, Model 2 has been deployed for further experimenting.

6. Deployment & Real-time Face Recognition

The system incorporates OpenCV's Haar cascades for face detection in real-time video streams. Frames from the live video stream are continuously captured, and the Haar cascades are applied to detect faces within each frame. The detected face images are preprocessed by resizing, converting to RGB, and normalizing pixel values. These preprocessed images are then fed into the trained models for face recognition. The models predict the class labels of the detected faces, and the recognized faces are displayed in the video stream with corresponding bounding boxes and class labels.



7. Experimentation and Refinement

The methodology is subject to experimentation and refinement, allowing for iterative analysis and performance improvements. Model hyperparameters and training strategies are fine-tuned to optimize the accuracy and efficiency of the face recognition system.

Some notable tweaks that happened through the course of experimentation include changes in the kernel size, add or removal of CNN layers, training for a greater or lesser number epochs, use of different criteria etc.

Of course, there is always room for improvement but even with this setup we managed to get some adequate results.

Conclusion

In this project, we developed a real-time face recognition system using deep learning techniques. The system successfully recognized faces in real-time video streams, demonstrating promising performance on a validation dataset. Three different models were designed and evaluated for face recognition tasks.

Based on the evaluation results, Model 2 achieved the highest accuracy of almost 75%, followed by Model 1 with an accuracy of 63%, and Model 3 with an accuracy of 36%. These models exhibited interesting performance results in terms of precision, recall, and F1-score, indicating their effectiveness in correctly identifying individuals from face images.

The experimental results also showcased the robustness of some models in contrast of others across different face classes, as evidenced by the confusion matrices. The high diagonal values in the confusion matrices indicate a strong ability to correctly classify face images, while the off-diagonal values represent misclassifications, providing insights into the models' strengths and areas for improvement.

The real-time face recognition system developed in this project holds great potential for various applications, such as security systems, access control mechanisms, and personalized user experiences. By accurately identifying individuals in real-time video streams, the system can enhance security protocols, streamline authentication processes, and deliver personalized services.

Moving forward, further research and development can be conducted to improve the system's performance and address potential challenges. This includes exploring advanced deep learning architectures, incorporating additional data augmentation techniques, and fine-tuning hyperparameters to optimize the models' accuracy and efficiency.

In conclusion, the real-time face recognition system presented in this project demonstrates the potential of deep learning techniques to achieve accurate and efficient face recognition in real-world scenarios. With ongoing advancements in deep learning and computer vision, this technology is expected to continue evolving and making significant contributions to various domains.