Project Report

Class Attendance using AI

Ubaid Ur Rehman

Supervised By: Prof. Michel Riveill, Diane Lingrand
Institution: University Côte d'Azur

Date: March 30, 2025

Contents

1 Introduction **6**
1.1 Motivation . . . 6
1.2 Objectives . . . 6

2 Methodology **7**
2.1 Development Environment and Tools . . . 7
2.2 Project Workflow . . . 7

3 Haar Cascade Classifier for face detection **8**
3.1 Face and Eye Detection Using Haar Cascades . . . 8
3.1.1 Grayscale Conversion . . . 8
3.1.2 Face Detection . . . 8
3.1.3 Eye Detection . . . 9
3.1.4 Face Validation . . . 9
3.1.5 Facial Region Extraction . . . 10

4 Machine Learning **10**
4.1 Preprocessing for Machine Learning: Create Dataset . . . 10
4.2 Preprocessing for Machine Learning: Wavelet Transform . . . 11
4.3 Machine Learning: SVM Model . . . 12
4.3.1 SVM Model Training and Evaluation . . . 12
4.3.2 Model Comparison with HalvingGridSearchCV . . . 12
4.3.3 Class Image Prediction with Best Model . . . 13
4.3.4 Results . . . 13

5 Deep Learning **14**
5.1 CNN Preprocessing Workflow . . . 14
5.1.1 Input Image Dimensions and Structure . . . 14
5.1.2 Automatic Feature Extraction and Pooling Techniques . . . 14
5.1.3 Managing Small Datasets and Preventing Overfitting . . . 14
5.2 CNN Model Implementation . . . 15
5.2.1 Initial Evaluation on Limited Dataset (20 Images per Class) . . . 15
5.2.2 Dataset Augmentation and Expanded Dataset . . . 15
5.2.3 Final Evaluation after Transformation . . . 15
5.3 Pre-Trained Models . . . 16
5.3.1 Loading the Data . . . 16
5.3.2 Applying VGG16 Model . . . 17
5.3.3 Applying InceptionV3 Model . . . 17
5.3.4 Applying ResNet50 Model . . . 17
5.3.5 Applying EfficientNetB0 Model . . . 17
5.3.6 Models Evaluation . . . 17
5.4 Model Evaluation and Comparison . . . 18
5.4.1 Model Selection for Image Classification . . . 18
5.4.2 Class Image Prediction with Best Pre-Trained Model . . . 18
5.4.3 Final Result . . . 19

6 Recognize the Gender of Each Face **19**
6.1 Dataset 19
6.2 Model Selection for Gender Classification 19
6.3 Training and Evaluation 20
6.4 Visualization of Model's Performance 20
6.5 Evaluation Metrics and Results 20
6.6 Predicting Gender on New Images 21
6.7 Result Summary 22

7 Ethical Considerations in AI for Attendance Systems **22**
7.1 Bias and Data Limitations 22
7.2 Computational and Hardware Constraints 22
7.3 Privacy and Ethical Concerns 23
7.4 Overall Constraints 23

List of Figures

1 Convert image to gray color 8
2 Draw rectangular around face 9
3 Draw the eyes 9
4 Crop the facial region of the image 10
5 Images get from class video 11
6 Wavelet Transform 12
7 Class Image 12
8 Predicted result 12
9 Class Image 13
10 Predicted result with best model 13
11 CNN Confusion Matrix 16
12 Class Image 18
13 Predicted result with best model 18
14 Gender accuracy loss plot 20
15 Confusion Matrix Gender 21
16 Class Image 21
17 Gender 1(female) 21
18 Gender 2 (Male) 22
19 Gender 3 (Male) 22

List of Tables

1 Model Comparison with Best Validation Accuracy Scores 13
2 Initial Evaluation Results on Limited Dataset (20 Images per Class) . . . 15
3 Final Evaluation Results After Transformation 16
4 Comparison of model performance on training and validation data. . . . 18

Listings

1 Face Detection Code 9

Abstract

Taking attendance in classrooms can be time-consuming and prone to errors when done manually. This project aims to simplify and automate the process using **Artificial Intelligence (AI)**. By analyzing classroom images, the system can **detect faces**, **identify students**, and even **classify gender**. Imagine a camera capturing a classroom photo, and within seconds, the system recognizes each student and marks attendance automatically—just like how face recognition is used to unlock smartphones. This project uses **Machine Learning (ML)** and **Deep Learning (DL)** techniques to make this possible, improving accuracy and reducing the need for manual attendance-taking.

Keywords: Automated Attendance, Machine Learning, Deep Learning, Face Detection, Student Identification, Gender Classification, AI, Education, Accuracy

1 Introduction

In educational institutions, attendance tracking is an essential but time-consuming process. Traditional methods rely on manual marking, which is prone to errors and inefficiencies. With advancements in **artificial intelligence**, particularly in **ML and DL**, automating attendance using **facial recognition** has become a viable solution. This project aims to develop an **AI-powered attendance system** that identifies students from classroom images and updates the attendance sheet accordingly.

The system consists of three key tasks: (1) **detecting and segmenting faces** in a classroom image, (2) **classifying the gender** of each detected face, and (3) **recognizing student identities** using ML and DL models. The ML approach incorporates **traditional feature extraction techniques** and classification models, while the DL approach employs **CNN-based architectures** and **Transfer Learning** for enhanced performance.

Ethical concerns such as **data privacy**, **potential biases**, and **computational limitations** are taken into account. This report provides an in-depth analysis of the **dataset creation**, **model training**, **evaluation metrics**, and **challenges encountered**.

1.1 Motivation

The motivation behind this project is to reduce the dependency on manual attendance systems, minimize human errors, and improve efficiency in academic institutions. Automated attendance tracking can save time, enhance accuracy, and provide a more seamless experience for both students and educators.

1.2 Objectives

The objective of this project is to develop a robust **ML and DL-based attendance tracking system** that can accurately recognize students from classroom images and automate the attendance marking process. The project focuses on:

- **Face Detection and Segmentation**: Using OpenCV-based approaches and DL models to accurately detect and segment faces in classroom images.

- **Student Identification**: Implementing ML and DL models to recognize individual students and mark their attendance.
- **Dataset Creation and Augmentation**: Developing a dataset from student images and videos while applying augmentations like zoom, flip, and crop to improve model generalization.
- **Performance Evaluation**: Assessing ML and DL models based on accuracy, loss, and other performance metrics to determine the best approach for automated attendance.
- **Gender Classification**: Training a CNN-based model on external datasets to classify student genders and improve recognition accuracy.
- **Addressing Ethical and Computational Constraints**: Ensuring data privacy, mitigating biases, and overcoming computational limitations while implementing AI-based attendance systems.

2 Methodology

To achieve the project objectives, a structured methodology is followed, incorporating various machine learning and deep learning techniques along with appropriate tools and technologies.

2.1 Development Environment and Tools

The project is implemented using **Python** as the primary programming language. The deep learning models are developed and trained using **TensorFlow** and **Keras**, while machine learning tasks utilize **Scikit-Learn**. The code is written and executed in **VS Code** on a system running **Windows 11** with an **RTX 3050 GPU**, enabling efficient model training.

2.2 Project Workflow

The methodology consists of the following key stages:

- **Data Collection and Preprocessing:** Gathering images from classroom videos and preparing the dataset by applying augmentation, grayscale conversion, and resizing techniques.
- **Face Detection and Segmentation:** Implementing **Haar Cascade Classifiers** to detect faces and eyes in classroom images, followed by facial region extraction.
- **Student Identification:** Utilizing **Support Vector Machines (SVM)** and deep learning-based **Convolutional Neural Networks (CNNs)** to classify student identities.
- **Attendance Sheet Generation:** Automating attendance marking based on detected faces and mapping them to the student database.

- **Deep Learning Model Implementation:** Training **custom CNN models** and evaluating pre-trained architectures such as **VGG16, InceptionV3, ResNet50, and EfficientNetB0** to enhance accuracy.
- **Performance Evaluation:** Assessing model effectiveness using metrics such as **accuracy, precision, recall, F1-score**, and visualizing results with confusion matrices.
- **Gender Classification:** Employing deep learning models to predict the gender of detected faces using external datasets for training.
- **Addressing Ethical and Computational Constraints:** Identifying potential biases, dataset limitations, and privacy concerns while optimizing computational resources to balance model accuracy and efficiency.

This structured approach ensures a robust, scalable, and efficient AI-based attendance system while addressing ethical considerations and hardware constraints.

3 Haar Cascade Classifier for face detection

3.1 Face and Eye Detection Using Haar Cascades

Facial recognition systems rely on accurate face detection as the initial step. This project uses the Haar Cascade Classifier from OpenCV [1] to detect faces and eyes in an image. The detection process follows these steps:

3.1.1 Grayscale Conversion

Convert the input image to grayscale for computational efficiency.

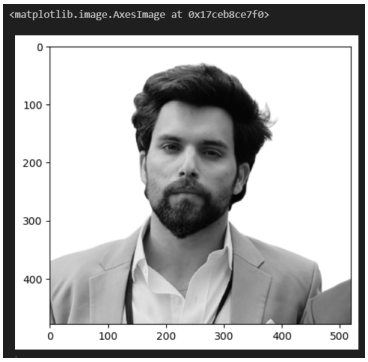

Figure 1: Convert image to gray color

3.1.2 Face Detection

Apply the pre-trained Haar Cascade classifier for face detection.

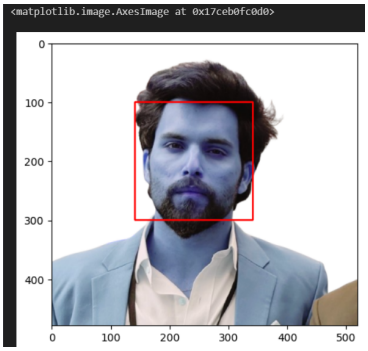

Figure 2: Draw rectangular around face

3.1.3 Eye Detection

If a face is detected, further analyze the region to detect eyes.

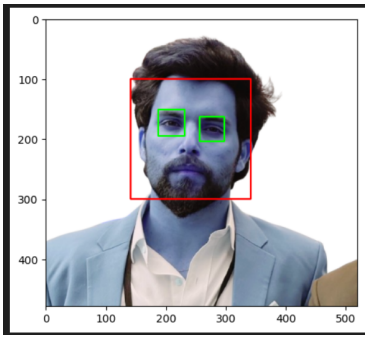

Figure 3: Draw the eyes

3.1.4 Face Validation

Retain the image only if two eyes are successfully detected[2].

```
def get_cropped_image_if_2_eye(image_path):
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Thanks to stackoverflow for this method to avoid false
       positive face detection
    face = face_cascade.detectMultiScale(
            gray,
            scaleFactor=1.2,
            minNeighbors=4,
            minSize=(50, 50),
            flags=cv2.CASCADE_SCALE_IMAGE
        )

    for (x, y, w, h) in face:
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray)

```

```
        if len(eyes) >= 2:
            return roi_color
        return roi_color
```

Listing 1: Face Detection Code

3.1.5 Facial Region Extraction

Extract the facial region for further processing in classification and identification tasks.

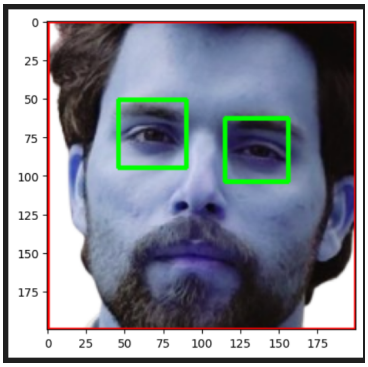

Figure 4: Crop the facial region of the image

This ensures that only clear and complete facial images proceed to the next stages of the attendance automation pipeline.

4 Machine Learning

4.1 Preprocessing for Machine Learning: Create Dataset

Before training the model, we preprocess the data by reading student images and extracting faces from a video[3] if needed, ensuring we have sufficient training data. We define paths for storing raw student images and cropped face images, which are organized in folders. The dataset consists of images from a class folder and a video taken by Ismail. We load all student folders from the dataset and store them in an array, with each folder corresponding to a student. Since some students have fewer images, we extract faces from the video to increase the dataset size. Using `OpenCV`, we detect and crop faces, manually assigning them to the correct student folders. After adding extracted faces to the student folders, we recount the number of images per student to confirm the dataset is sufficiently populated. A separate folder named `cropped` is created to store preprocessed face images, and any existing cropped images are deleted before regenerating them. We define a method to detect faces and save cropped images, which are stored in the `cropped` folder for further processing. After saving the cropped images, we load all folders to check the processed images before moving to model training. To ensure dataset quality, we manually review and delete unwanted images from the `cropped` folder, leaving only the final images for training. If more data is needed, we apply transformations such as rotation, flipping, and zooming to increase image diversity, which enhances the model's robustness. A dictionary is created to map student names to numerical class labels, preparing the

data for training the classification model. The preprocessed images, along with wavelet-transformed images, are used to train the model, ensuring the dataset is well-structured for optimal performance.

```
Folder: abhinav, Images: 858
Folder: Algassimou DIALLO, Images: 64
Folder: Eli, Images: 2
Folder: gerid, Images: 554
Folder: hadi, Images: 70
Folder: hasham, Images: 329
Folder: hassan, Images: 290
Folder: Haxhiu Arion, Images: 438
Folder: imtiaz, Images: 76
Folder: ismail, Images: 155
Folder: Juliana, Images: 534
Folder: mellisa, Images: 498
Folder: Nabila, Images: 912
Folder: nestor, Images: 6
Folder: Tuan Anh NGUYEN, Images: 84
Folder: Tuan HOANG, Images: 998
Folder: ubaid, Images: 294
Folder: usman, Images: 321
Folder: yousef, Images: 216
Folder: zahid, Images: 17
Folder: zeev, Images: 183
```

Figure 5: Images get from class video

4.2 Preprocessing for Machine Learning: Wavelet Transform

Wavelet-transformed images [4] enhance edge detection, highlighting key facial features such as eyes, nose, and lips. This transformation improves feature representation for training. Wavelet transformation extracts high-frequency details, preserving structural information while enhancing edges and contours, making it useful for face recognition. To improve feature extraction, the original image is combined with its wavelet-transformed counterpart, providing the model with diverse information and boosting classification accuracy. All images are resized to a fixed shape to ensure uniform input, and standardization or normalization can be applied to maintain feature scale consistency.

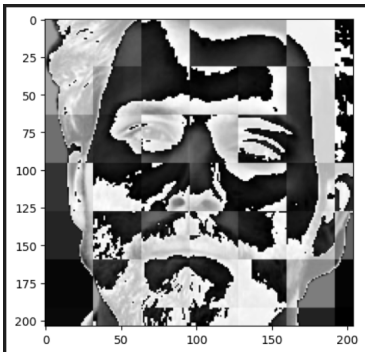

Figure 6: Wavelet Transform

4.3 Machine Learning: SVM Model

4.3.1 SVM Model Training and Evaluation

Data cleaning and preprocessing are critical steps to ensure that the data is ready for model training. The SVM model, using an RBF kernel, is trained with a pipeline that scales the data. After training, the model is evaluated using classification reports and confusion matrices to assess its performance. Once the evaluation is complete, the trained model and the class dictionary are saved for later use. In the final step, the trained model is used to predict the classes of faces in images, and the results are displayed with bounding boxes and labels for visual identification.

Figure 7: Class Image

Figure 8: Predicted result

These are not good results; let's try to improve them.

4.3.2 Model Comparison with HalvingGridSearchCV

The goal of using HalvingGridSearchCV is to fine-tune the model for better performance. Several models, including SVM, Random Forest, and Logistic Regression, are tested with their respective parameters. To make the grid search process more efficient, PCA is applied to reduce the feature space from 4096 to 500. Grid search is performed for each model with various parameters, and HalvingGridSearchCV is used to improve the model's performance. The best-performing model is selected based on the results of the grid search process.

4.3.3 Class Image Prediction with Best Model

Once the best model has been selected from the comparison process, it is used to predict class images. The final model is evaluated to ensure that it performs accurately on the test set. This model is then used to predict the classes of faces detected in class images, with results displayed using bounding boxes and labels to mark the predicted classes.

Figure 9: Class Image

Figure 10: Predicted result with best model

4.3.4 Results

The machine learning models were trained and evaluated on the dataset of student images and class videos. Initially, the dataset was created using a combination of student images and class videos. OpenCV's haar-cascades was used for face and eye detection, providing a reliable method for feature extraction. Although the dataset initially lacked sufficient images, video was used to capture more faces, ensuring a robust dataset for training. Image transformation techniques such as zoom, flip, and crop were applied to augment the dataset as needed. Additionally, a wavelet transform was applied to detect image edges, making facial features like eyes, nose, and lips more distinguishable.

The images were then combined into a feature vector with a size of 4096 by merging both the original cropped images and wavelet-transformed images. Initially, an SVM model was tested, which provided decent results, but it showed limitations with accuracy, correctly predicting only 3 to 4 students. Subsequently, different models were tested, including SVM, RandomForestClassifier, and LogisticRegression.

The results for each model were as follows:

Model	**Best Score**
SVM	74%
Random Forest	78%
Logistic Regression	86%

Table 1: Model Comparison with Best Validation Accuracy Scores

Based on these results, Logistic Regression emerged as the best-performing model, significantly outperforming both SVM and Random Forest. Despite its improved accuracy, it still requires further refinement to optimize its performance.Finally, the prediction is not 100% accurate because we need a larger dataset. If our dataset is captured from each angle and used for training, then any kind of class image will be predicted correctly.

5 Deep Learning

5.1 CNN Preprocessing Workflow

In this section, we outline the preprocessing steps used for building a Convolutional Neural Network (CNN) model on a dataset of student images. CNNs are powerful tools for image classification tasks, leveraging the spatial structure of images directly without requiring manual feature extraction. This allows for more efficient learning of patterns, such as edges and textures, making CNNs particularly effective for tasks like image classification.

5.1.1 Input Image Dimensions and Structure

For this CNN model, we use the original full-color images with dimensions of $32 \times 32 \times 3$, preserving their inherent spatial structure. This is in contrast to traditional machine learning approaches, where images are often flattened into 1D arrays, which can lose crucial spatial information. By keeping the image in its original shape, we allow the CNN to learn important features directly from the 2D structure, such as edges, corners, and other visual patterns.

5.1.2 Automatic Feature Extraction and Pooling Techniques

In a CNN, the convolutional layers automatically extract features from the images, such as edges, textures, and basic shapes, through the use of filters. This eliminates the need for manual feature engineering methods, such as wavelet transformations, which are typically used in traditional machine learning methods. Instead, CNNs learn these features during training, making the process more efficient.

We also apply padding to the image borders to maintain the original image dimensions after applying convolutions. Pooling techniques, such as max pooling or average pooling, are used to reduce the spatial dimensions of the image, helping the model focus on the most important features while retaining computational efficiency. These pooling techniques downsample the image, which improves performance by reducing the complexity of the data without losing critical information.

5.1.3 Managing Small Datasets and Preventing Overfitting

The dataset for this model consists of an average of 20 images per student, making it relatively small. To mitigate the risk of overfitting, we optimize the model by choosing appropriate pooling techniques and adjusting the network architecture to avoid excessive complexity. This ensures the model generalizes well to unseen data. Additionally, the dataset is split into training and test sets, with 80% of the data used for training and 20% reserved for testing. This train-test split helps evaluate the model's performance and ensures it can generalize effectively to new data.

5.2 CNN Model Implementation

5.2.1 Initial Evaluation on Limited Dataset (20 Images per Class)

Before applying any image transformations, the CNN model was trained on a limited dataset containing 20 images per class. The model exhibited severe overfitting due to the small dataset size. Although the training accuracy was relatively high, the validation accuracy suffered significantly. The results are as follows:

Metric	**Value**
Training Loss	2.8873
Training Accuracy	0.9849
Validation Loss	10.1384
Validation Accuracy	0.0723

Table 2: Initial Evaluation Results on Limited Dataset (20 Images per Class)

As we can see from the metrics above, the model is overfitting on the training set, achieving high accuracy on the training data but failing to generalize to the validation set. The validation loss is extremely high, and the validation accuracy is very low, highlighting the issue of overfitting due to the limited dataset.

5.2.2 Dataset Augmentation and Expanded Dataset

To mitigate overfitting, several image transformation techniques were applied to augment the dataset. These included image zoom, rotation, flipping, brightness and contrast adjustments, Gaussian noise addition, and blurring[5]. The application of these transformations expanded the dataset and introduced diversity in the images, making the model more robust to various real-world variations.

The transformed dataset allowed the model to train on a much larger and varied set of images, which helped reduce overfitting and improve performance on the validation set.

5.2.3 Final Evaluation after Transformation

After applying the transformations and retraining the CNN model on the expanded dataset, the model exhibited significant improvements in validation performance. Overfitting was successfully reduced, and the model achieved near-perfect accuracy on both the training and validation sets. The final results are as follows:

Metric	Value
Training Loss	0.0954
Training Accuracy	1.0000
Validation Loss	0.0926
Validation Accuracy	0.9960

Table 3: Final Evaluation Results After Transformation

As shown by the final metrics, the validation accuracy is now closely aligned with the training accuracy, indicating that the model generalizes well to unseen data. The validation loss has dropped significantly, and the validation accuracy has improved to nearly 99.6%. The application of dataset augmentation techniques has effectively resolved the overfitting issue, resulting in a well-performing model. The confusion matrix is as follows:

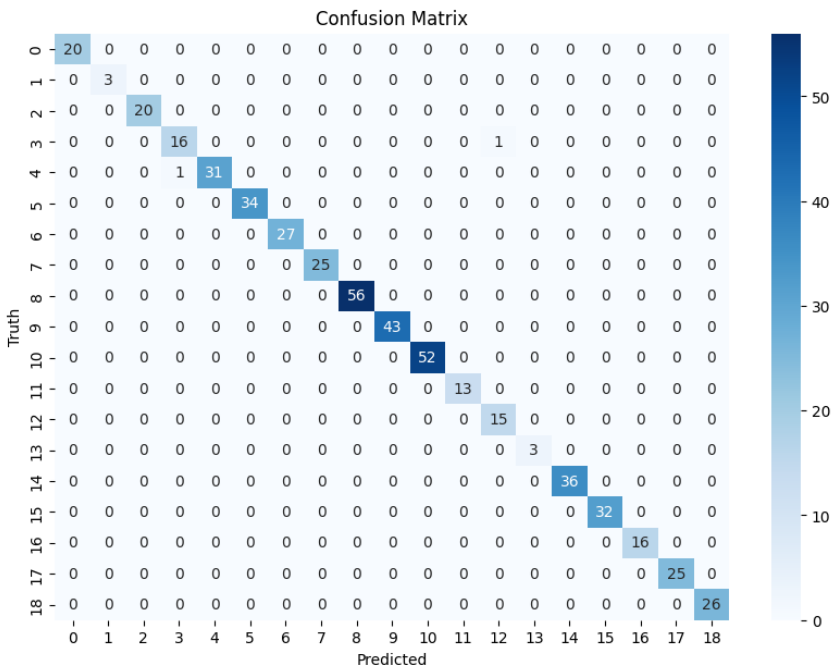

Figure 11: CNN Confusion Matrix

However, the class prediction results are still not fully satisfactory. We will explore deeper learning Pre-Trained models to further improve our results.

5.3 Pre-Trained Models

Next, we will evaluate four top pre-trained models [6] step by step to compare their performance on our dataset.

5.3.1 Loading the Data

Before applying pre-trained models, we first **load and preprocess the dataset**. The dataset consists of multiple images categorized into different classes. We **resize all**

images to a uniform size and **normalize the pixel values**. The data is then **split into training and testing sets** to evaluate model performance effectively. This ensures that all models work on the same dataset, making the results comparable.

5.3.2 Applying VGG16 Model

The **VGG16 model**, known for its deep architecture with 16 layers, is applied first. We **load the pre-trained VGG16 model without its top layers** and freeze its weights. A **custom classification head with fully connected layers** is added on top. The model is trained using **categorical cross-entropy loss** and **RMSprop optimizer**. After training for **15 epochs**, the model is saved, and predictions are made on test images. The performance is recorded for further comparison.

5.3.3 Applying InceptionV3 Model

Next, we apply the **InceptionV3 model**, which has an **inception module** that improves efficiency by using multiple kernel sizes. We **load the base model without the top layers** and add **custom dense layers**. **Batch normalization** and **dropout layers** are included to prevent overfitting. The **Adam optimizer** is used for training, and the model is fine-tuned for **15 epochs**. After training, the model is saved, and predictions are generated on test images. Performance metrics like **accuracy** are noted for evaluation.

5.3.4 Applying ResNet50 Model

The **ResNet50 model** is implemented next, which introduces **residual connections** to prevent vanishing gradients. The model is **loaded without its classification head**, and additional **fully connected layers** are added. A **global average pooling layer** is used to maintain efficiency. The model is trained for **20 epochs** using the **Adam optimizer**. Once trained, predictions are made on test images, and the model's **accuracy is compared** with previous models.

5.3.5 Applying EfficientNetB0 Model

Finally, the **EfficientNetB0 model** is applied. This model uses **compound scaling** to balance **width, depth, and resolution** for optimal performance. We **load the model without the top layers** and add a **fully connected layer with dropout**. The model is trained using the **Adam optimizer for 20 epochs**. After saving the model, test images are used for predictions, and its **accuracy is recorded**. This model is expected to perform well due to its **optimized architecture**.

5.3.6 Models Evaluation

After training all four models, their performance is evaluated based on **accuracy, loss, and prediction efficiency**. **VGG16** provides stable results but **requires more parameters**. **InceptionV3** improves efficiency but **may not generalize** as well in some cases. **ResNet50** avoids vanishing gradients and **achieves high accuracy**. **EfficientNetB0**, with its **optimized architecture**, balances **accuracy and computational cost**. Based on the results, the most suitable model for the given dataset is chosen for **deployment**.

Model	Train Loss	Train Accuracy	Val Loss	Val Accuracy
VGG16	0.0033	100.00%	0.0206	99.19%
InceptionV3	0.0315	99.39%	0.1339	96.16%
ResNet50	0.4501	90.79%	0.5955	86.67%
EfficientNetB0	0.0102	99.65%	0.0235	99.19%

Table 4: Comparison of model performance on training and validation data.

5.4 Model Evaluation and Comparison

5.4.1 Model Selection for Image Classification

- **VGG16** and **EfficientNetB0** both show excellent performance with validation accuracy of **99.19%**, making them strong choices.
- **InceptionV3** has a slightly lower validation accuracy (**96.16%**) but might be useful if the dataset is small due to its efficient architecture.
- **ResNet50** has the lowest accuracy (**86.67%**), making it a weaker choice for this dataset.
- **EfficientNetB0** offers a balance of high accuracy and efficiency, making it a recommended choice for real-world applications.

5.4.2 Class Image Prediction with Best Pre-Trained Model

So, we are selecting VGG16 as the best model in this case, and we also saved the attendance sheet with the help of this model. Now, let's take another student image with three students and see if the model predicts correctly. However, this model still needs improvement to accurately predict all faces in the class. Below is the figure of the class and its prediction.

Figure 12: Class Image

Figure 13: Predicted result with best model

The above results are correct in this case, as the faces were identified accurately. However, these images were taken with an iPhone and are clear. If the images are blurry or not clear, especially in a classroom setting, the model may not produce the expected output.

5.4.3 Final Result

Based on the evaluation, **VGG16** and **EfficientNetB0** demonstrate the highest validation accuracy (**99.19%**), making them the best candidates. However, further improvements are needed for handling unclear or low-quality images in real-world scenarios.

6 Recognize the Gender of Each Face

In this section, we aimed to build a gender classifier that could detect the gender of faces from images. We used a gender classification dataset[7], split into training, validation, and test sets, to train and evaluate the model.

6.1 Dataset

The dataset consisted of cropped images of males and females, which were split into the following directories:

- **Training Dataset**: 23,000 images of each class (male and female).
- **Validation Dataset**: 5,500 images of each class.

These images were used for training and evaluating the gender classification model.

6.2 Model Selection for Gender Classification

To detect gender, we used a Convolutional Neural Network (CNN) model with several convolutional layers, max-pooling layers, and dropout layers for regularization[8]. This model was designed to process images of size 32x32 pixels with three color channels (RGB). It was built using Keras, a deep learning framework, with the following architecture:

- **Conv2D Layer** with 32 filters and 3x3 kernel size (followed by max pooling).
- **Conv2D Layer** with 64 filters and 3x3 kernel size (followed by batch normalization and max pooling).
- **Conv2D Layer** with 128 filters and 3x3 kernel size (followed by batch normalization and max pooling).
- **Dense Layer** with 128 units for further processing and classification into two categories (Male and Female).
- **Output Layer** with 1 unit and a sigmoid activation function to classify the gender.

6.3 Training and Evaluation

We trained the CNN model using the training dataset with a batch size of 512 and for 30 epochs. The model's performance was evaluated using the validation dataset. After training, the model was saved and evaluated on the test dataset, providing us with the final test accuracy and loss. The test accuracy achieved was **92%**, demonstrating good performance on gender classification tasks.

6.4 Visualization of Model's Performance

During training, the model's loss and accuracy were monitored. We plotted the training and validation loss and accuracy curves to visualize the performance and identify any signs of overfitting or underfitting.

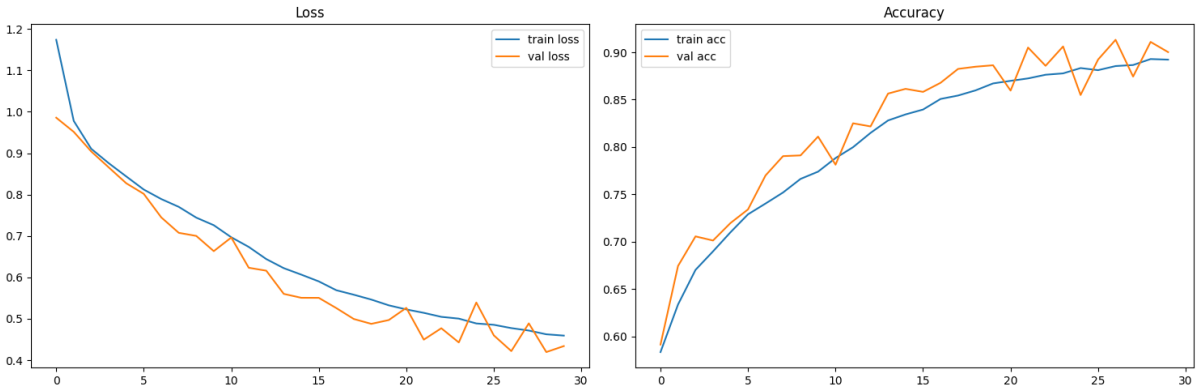

Figure 14: Gender accuracy loss plot

6.5 Evaluation Metrics and Results

The model's performance was evaluated using a confusion matrix and classification report, which provided a detailed analysis of the model's accuracy, precision, recall, and F1-score. The confusion matrix allowed us to understand how well the model performed on both male and female images, with **92% accuracy** achieved on the test data.

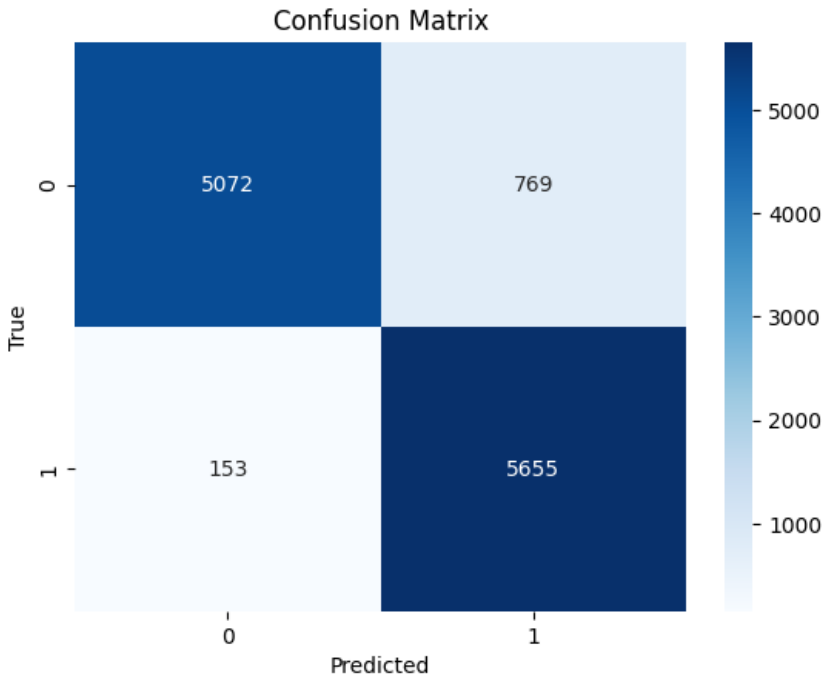

Figure 15: Confusion Matrix Gender

6.6 Predicting Gender on New Images

We implemented methods to predict the gender of new images using the trained model. This included predicting gender for a batch of images and for any individual image provided to the model. For instance, a test image might show a "Male" label with a confidence score of **92%**.

Figure 16: Class Image

Figure 17: Gender 1(female)

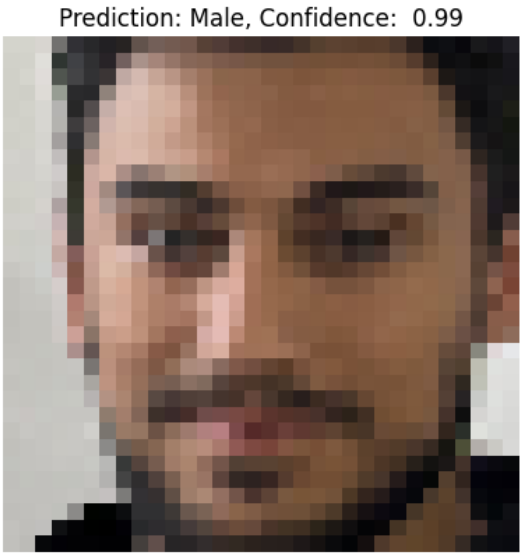

Figure 18: Gender 2 (Male)

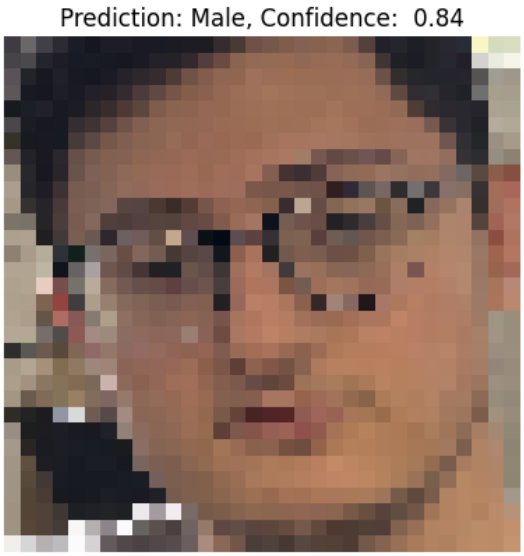

Figure 19: Gender 3 (Male)

6.7 Result Summary

In conclusion, the gender detection model achieved **92% accuracy** on the training set and similar results on the test set. The dataset used played a critical role in obtaining these results. However, further improvements in model architecture and data augmentation may help achieve better performance.

7 Ethical Considerations in AI for Attendance Systems

7.1 Bias and Data Limitations

AI-based attendance systems rely on data quality and diversity. A limited dataset can lead to poor generalization, reducing model performance in real-world scenarios. Gender imbalance is another concern, as the dataset contains only three girls, potentially affecting face detection accuracy. Additionally, external datasets, such as those from Kaggle, were used for gender classification, but they may not reflect the actual distribution of students. Furthermore, the cropped face images from classroom videos may lack clarity, leading to inaccurate predictions.

7.2 Computational and Hardware Constraints

Training deep learning models requires substantial computational resources, which can be a limiting factor. Due to hardware constraints, the model was trained on low-resolution (128-pixel) images, impacting accuracy. While higher resolution could enhance performance, it is not feasible with the available system. These limitations affect both model efficiency and its ability to generalize well to real-world conditions.

7.3 Privacy and Ethical Concerns

AI-based attendance tracking raises significant ethical concerns. Capturing student images without explicit consent introduces privacy and legal issues. Continuous monitoring can also lead to discomfort and trust concerns, making students feel observed. Additionally, fairness in AI is crucial—misclassification may result in false absenteeism, unfairly affecting students. Algorithmic biases, stemming from imbalanced datasets, can further lead to discrimination, making ethical AI deployment a priority.

7.4 Overall Constraints

Several factors impact the effectiveness of AI-driven attendance systems. A small dataset and gender imbalance limit classification accuracy, while computational constraints prevent high-resolution training. Cropped images from classroom videos may lack clarity, affecting recognition reliability. Most importantly, privacy concerns and ethical considerations regarding consent must be carefully addressed before widespread adoption of such systems.

References

[1] OpenCV. *Haar cascade for face detection*. Accessed: Mar 30, 2025. n.d. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html?highlight=haar.

[2] StackOverflow. *Limiting the number of detected faces*. Accessed: Mar 30, 2025. 2018. URL: https://stackoverflow.com/questions/52288066/how-to-limit-number-of-faces-detected-by-haar-cascades.

[3] StackOverflow. *Detecting faces in video, cropping, and saving frames*. Accessed: Mar 30, 2025. 2020. URL: https://stackoverflow.com/questions/60546695/detect-faces-in-video-crop-them-and-save-all-frames-in-the-same-order-opencv.

[4] StackOverflow. *Combining PyWavelets and OpenCV for image processing*. Accessed: Mar 30, 2025. 2014. URL: https://stackoverflow.com/questions/24536552/how-to-combine-pywavelet-and-opencv-for-image-processing.

[5] StackOverflow. *Rotating an image using OpenCV*. Accessed: Mar 30, 2025. 2012. URL: https://stackoverflow.com/questions/9041681/opencv-python-rotate-image-by-x-degrees-around-specific-point.

[6] Analytics Vidhya. *Top 4 pre-trained models for image classification*. Accessed: Mar 30, 2025. 2020. URL: https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/.

[7] Kaggle. *Gender classification dataset*. Accessed: Mar 30, 2025. n.d. URL: https://www.kaggle.com/datasets/cashutosh/gender-classification-dataset/code.

[8] kaggle. *Gender classification using Keras*. Accessed: Mar 30, 2025. n.d. URL: https://www.kaggle.com/code/ubaidurrehman404/gender-classification-w-keras/edit.

[9] Tom Kaminski. *HalvingRandomSearchCV vs GridSearchCV*. Accessed: Mar 30, 2025. 2020. URL: https://medium.com/@tom.kaminski01/halvingrandomsearchcv-vs-gridsearchcv-f1cd42f54b3b.

[10] Analytics Vidhya. *Learning Image Classification with CNN*. Accessed: Mar 30, 2025. 2020. URL: https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/.