DA 526: Image Processing with Machine Learning

Project Report
Face Recognition-based Attendance System

Guided By

Prof. Debanga Raj Neog

Team Members:

Harshil Sadharakiya	**224101022**
Kaushal Mistry	**224101031**
Utsav Bheda	**224101054**
Vatsal Vasoya	**224101056**
Chinmay Rathod	**224101063**

Table of Contents

Introduction to Problem Statement

In today's fast-paced world, managing attendance in educational institutions and workplaces is a critical task. The traditional methods of attendance management, such as manual register entry or using barcode or RFID cards, are time-consuming and prone to errors. With the advancements in computer vision technology, face recognition-based attendance systems have emerged as a reliable and efficient solution to tackle this issue.

The face recognition-based attendance system is a biometric technology that uses facial features to verify and authenticate the identity of an individual. This technology has gained immense popularity due to its ability to accurately identify individuals, even in large groups, with high speed and convenience. In this project, we aim to develop a face recognition-based attendance system that will automate the attendance management process in educational institutions and workplaces.

The proposed system will use a camera to capture the images of individuals, and the face recognition algorithm will identify and verify their identity. The system will then mark the attendance of the identified individual in a database.

This report presents the details of the development of the face recognition-based attendance system. It provides an overview of the system's architecture, working, and implementation details. The report also presents the results of the system's performance evaluation, which includes accuracy, speed, and efficiency. Finally, the report concludes with the system's potential applications and future directions.

Related Works

- Eigenface is a popular approach to face recognition that uses Principal Component Analysis (PCA) for feature extraction.
- Eigenface works by representing a set of face images as a high-dimensional vector space, with each image represented as a point in that space. PCA is then used to extract the most important features or components from the image set, which are represented as eigenvectors of the covariance matrix of the image set.
- To perform face recognition using the Eigenface approach, a set of training images is first used to compute the eigenvectors. Each training image is represented as a linear combination of the eigenvectors, with the weights of the linear combination forming a feature vector for that image. These feature vectors are then used to train a classifier, such as a Support Vector Machine (SVM), to recognize different individuals.
- During testing, a test image is first projected onto the eigenvectors to obtain its feature vector. The classifier is then used to determine the identity of the person in the test image based on the similarity between its feature vector and those of the training images.

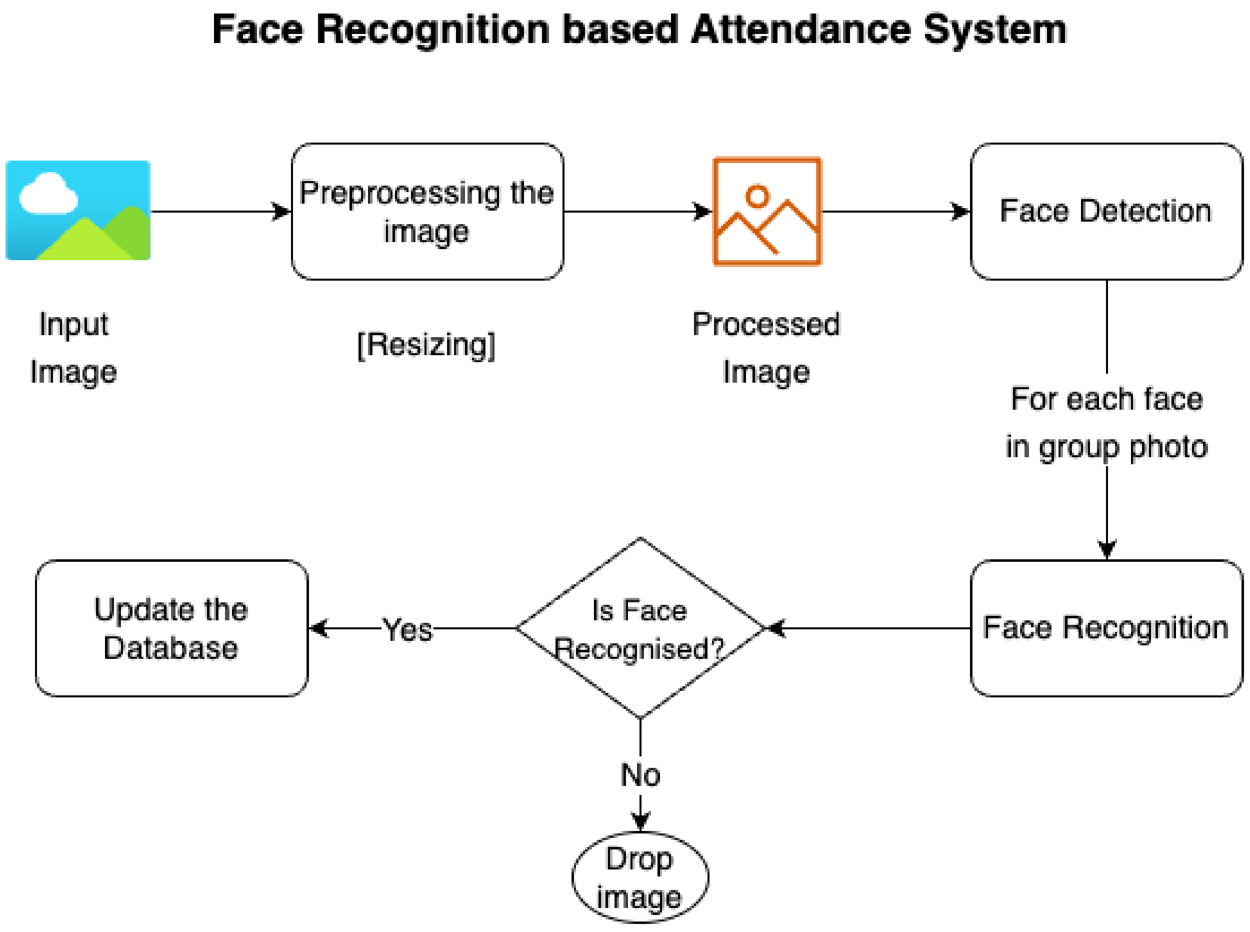
DataSet

The dataset used in the development of the Face Recognition-based Attendance System plays a crucial role in the system's accuracy and performance. The quality and size of the dataset determine the robustness of the face recognition algorithm and the system's ability to identify individuals accurately.
For that, initially, we chose the Bollywood celebrity faces dataset.

- Link to dataset: [Bollywood Celebrity Faces | Kaggle]
- The dataset has the below details:
  - 100 celebrity classes (person).
  - Each class has around 100~120 images.
  - Each image contains only one face.
  - Each image is associated with only one class.

Flow Diagram

The overall flow of the project is depicted in the flowchart below:



[Image 1: Flow Chart]

Methods

The project is divided into 4 modules. The modules are as follows:

1. Face Detection

The face detection section of the Face Recognition-based Attendance System plays a crucial role in the system's overall performance. The face detection module is responsible for locating faces in the images captured by the camera, which is the first step toward identifying and verifying an individual's identity.

For the face detection module, we have tried different methods below:

- OpenCV HaarCascades
  The HaarCascade algorithm is based on the Viola-Jones framework, which utilizes machine-learning techniques to detect objects in images. It employs a set of Haar-like features, which are rectangular filters that capture various patterns of contrast in the image. These features act as classifiers to identify regions of interest that may contain faces.
  During the face detection process, the HaarCascade algorithm scans the image using a sliding window technique, examining each region at multiple scales. At each scale, the algorithm applies the Haar-like features to evaluate whether the region resembles a face or not. If a region is classified as a potential face, it is further analyzed to refine the detection and eliminate false positives.
  The HaarCascade algorithm is known for its fast processing speed, making it suitable for real-time applications. Additionally, it provides a balance between accuracy and efficiency, making it a popular choice for face-detection tasks.

- HOG-based face detection
  The HOG-based face detection method relies on capturing the local gradient information within an image. It extracts features based on the distribution of gradient orientations, which are computed using edge detection algorithms like the Sobel operator. By considering the orientation of gradients, the HOG algorithm effectively captures the structure and texture information of facial regions.
  The process of HOG-based face detection involves the following key steps:
  - Image Preprocessing
  - Gradient Computation
  - Cell Formation
  - Block Normalization
  - Training and Detection

  It has demonstrated high accuracy and reliability in various face detection scenarios. It can handle complex lighting conditions, scale variations, and partial occlusions. However, it may require higher computational resources compared to HaarCascade due to the complexity of feature extraction and classification.

- YOLO deep learning-based model
  YOLO (You Only Look Once) is an object detection algorithm that can be utilized for face detection tasks. Unlike traditional approaches that use sliding windows, YOLO is a single-shot

detection algorithm, meaning it performs object detection in a single pass over the input image.
The YOLO algorithm divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell. These bounding boxes represent the regions where objects, including faces, are likely to be present. YOLO can detect multiple objects, including faces, within a single image.

  - The key steps involved in YOLO face detection are as follows:
    Input Processing: The input image is resized to a fixed size to fit the YOLO network's input requirements.
  - Feature Extraction: The resized image is passed through a deep convolutional neural network (CNN) to extract features at multiple scales. These features capture various characteristics and patterns present in the image, including facial features.
  - Grid Formation: The image is divided into a grid of cells, typically with a size of, for example, 13x13 or 19x19.
  - Bounding Box Prediction: For each grid cell, YOLO predicts bounding boxes by regressing coordinates relative to the cell's position. Each bounding box includes the coordinates of the box's top-left corner, width, height, and a confidence score representing the likelihood of containing an object, such as a face.
  - Non-Maximum Suppression: To eliminate redundant and overlapping bounding boxes, a non-maximum suppression algorithm is applied. It removes duplicate or highly overlapping boxes with lower confidence scores, keeping only the most relevant and accurate predictions.
  - Face Classification: YOLO assigns class probabilities to the remaining bounding boxes. In the case of face detection, it identifies whether the detected object is a face or belongs to another class.

YOLO is known for its real-time performance and ability to detect objects, including faces, with remarkable accuracy. It can handle object detection in complex scenes with multiple objects, occlusions, and varying scales. However, YOLO's speed and efficiency come at the expense of slightly reduced accuracy compared to slower, multi-stage detection algorithms.
When incorporating YOLO for face detection in the Face Recognition-based Attendance System, it offers the advantage of fast and accurate face localization. The detected faces can then be passed on to the subsequent face recognition module for identity verification and attendance marking.

- MTCNN (Multi-task cascaded convolution network)
  MTCNN stands for Multi-task Cascaded Convolutional Networks. It is a deep learning-based face detection algorithm that is widely used in computer vision applications, particularly in face detection and facial feature alignment.
  MTCNN is known for its ability to efficiently detect faces in images with high accuracy. It uses a cascaded architecture consisting of three neural networks: the Proposal Network (P-Net), the Refine Network (R-Net), and the Output Network (O-Net).
  Here is a brief overview of how MTCNN works:

  - P-Net: The first stage of MTCNN is the Proposal Network. It scans the image at multiple scales and locations, generating a set of candidate bounding boxes that may contain faces. It also produces a probability score for each bounding box, indicating the likelihood of a face being present.

- R-Net: The second stage is the Refine Network. It takes the candidate bounding boxes generated by the P-Net and refines them by eliminating overlapping boxes and accurately aligning the bounding boxes with the faces. It also performs a more accurate classification of whether the detected region contains a face or not.
- O-Net: The third and final stage is the Output Network. It further refines the bounding boxes and performs facial landmark detection, identifying key facial features such as the eyes, nose, and mouth. It also provides a final face/non-face classification.

MTCNN has gained popularity due to its ability to detect faces accurately, even under challenging conditions such as variations in lighting, pose, and occlusions. It has been widely used in various applications, including facial recognition, emotion analysis, and facial attribute detection.

Overall, MTCNN plays a crucial role in face detection tasks, providing a robust and efficient solution for identifying and localizing faces in images and videos.

2. Preprocessing for face classification

- After detection we have to process each image to feed into a face recognizer model.
- For that we are resizing the image into 224x224 and we are normalizing it for faster training and making all images into the same scale.
- For normalizing we are changing the image such that for all channel mean and standard deviation is 0.5.

3. Face Recognition

- VGG
    - VGG face is vgg16 architecture trained on VGG face dataset.
    - It contains 138 Million parameters.
    - Architecture contains two parts, feature extractions, and face classification.
    - Feature extraction extracts 25088 features, which is very large, so we have frozen all layers except the last layer. So only the last layer's weights will be changed while training.
    - The architecture of the model is very simple but contains a large number of parameters.
    - So the model takes more time than other neural network models which contain fewer parameters and are computationally less expensive than VGG.
- Inception ResNet V1
    - Inception ResNet V1 uses the inception module which was introduced in the GoogleLeNet architecture developed by Google.
    - Then it combines this inception module with the concept of residual connections(skip connections) which was introduced in the ResNet architecture to mitigate the issue of vanishing gradient in very deep models.
    - Here we are using the idea of transfer learning by using a pre-trained deep mode on the VGGFace2 dataset which contains 2.6 million images having more than 2600 classes.

    - This mode architecture contains 107 Million trainable parameters.
    - To fine-tune the model we are freezing the layers till the Convolution blocks, which do the task of extracting features from the face images. So these weights will not be updated during the training.
    - We are modifying the architecture of the last linear layer according to the requirements of our classification task based on our dataset.
    - After modifying the model architecture it will have 2,42,276 learnable parameters with 100 classes in the classification problem.

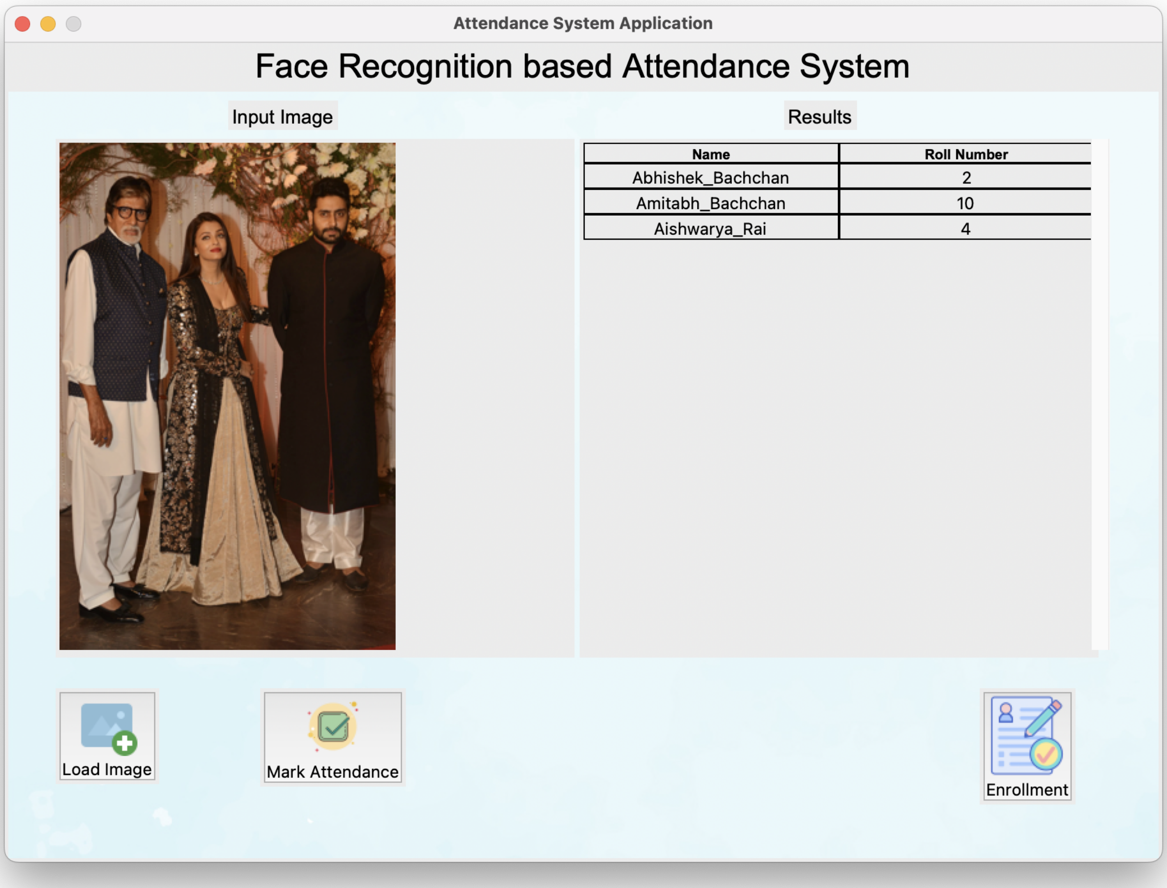
4. Attendance Recording

- We are maintaining an Excel sheet of all students to mark their attendance.
- To mark the attendance a photo of an individual or group of students will be uploaded in the system.
- From the uploaded image, all the faces of individuals will be detected by the system.
- Then using face recognition, our system will map each individual face detected with a face of a student from our list of all students.
- For all the students recognized in the image, attendance will be marked in the attendance sheet.

UI for accessing the project

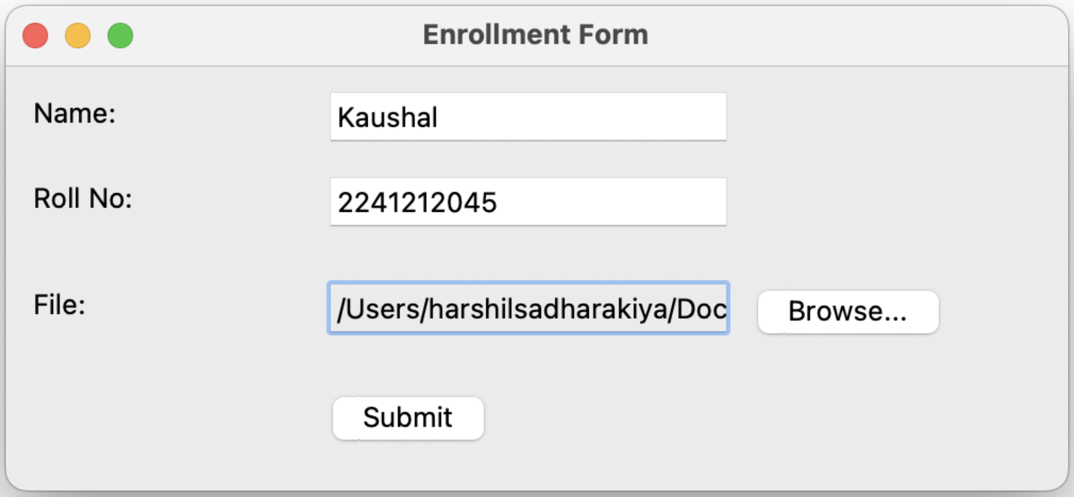
We have created the UI for smooth access to the system. It allows the users to add the group photo of the students by clicking the “Load Image” button and the preview of the image would be shown in the “Input Image” section. Then to mark the attendance of the people in the input image click “Mark Attendance”. It will detect the people from the input image and show the names and roll numbers of the people from the image in the Results section.

To add the new student to the dataset, we can click on the “Enrollment” button and it will open the pop-up shown below. To add a new student, we require some set of images of the new student and his/her basic details. Provide the path of the folder containing the images. Upon submitting, it will add the images in the directory of the name provided in the dataset location and re-train the model to detect the newly added student.


Attendance System Application
Face Recognition based Attendance System
Input Image
Results
Load Image
Mark Attendance
Enrollment

Name	Roll Number
Abhishek_Bachchan	2
Amitabh_Bachchan	10
Aishwarya_Rai	4

[Home page of the UI]



Name:	Kaushal
Roll No:	2241212045
File:	/Users/harshilsadharakiya/Doc

[Enrollment form for adding new student]

Experiments & Results

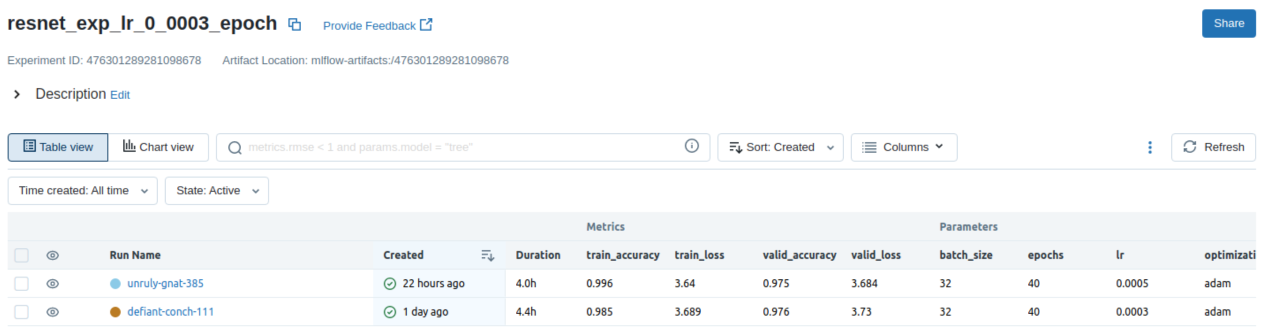
Face Detection:

We have tried the different face detection methods and the results on a set of group photos. Based on the experiments and results we found that HaarCascade and HOG-based face detectors are more prone to false positives and are less accurate. YOLO and MTCNN are good in terms of accuracy. MTCNN is computationally expensive as it uses more layered CNN. YOLO is fast. So we have used YOLO while training as the data set is clean. We have used MTCNN while testing for more accuracy.

Face Recognition:

ResNet

	Precision	Recall	F1-Score
Accuracy			0.97
Macro Average	0.97	0.98	0.97
Weighted Average	0.98	0.97	0.97

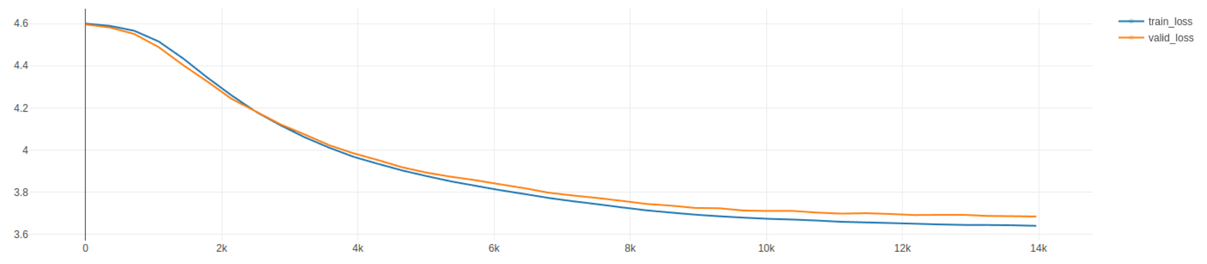


resnet_exp_lr_0_0003_epoch

Experiment ID: 476301289281098678 Artifact Location: mlflow-artifacts:/476301289281098678

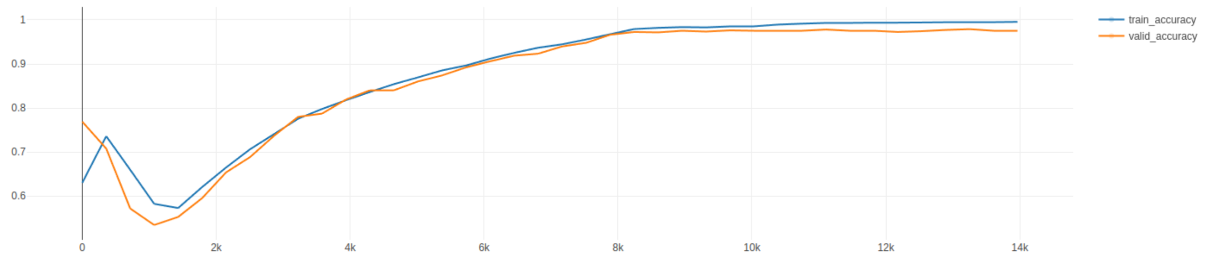
				Metrics				Parameters			
Run Name	Created	Duration	train_accuracy	train_loss	valid_accuracy	valid_loss	batch_size	epochs	lr	optimizati
unruly-gnat-385	22 hours ago	4.0h	0.996	3.64	0.975	3.684	32	40	0.0005	adam
defiant-conch-111	1 day ago	4.4h	0.985	3.689	0.976	3.73	32	40	0.0003	adam

ResNet mlflow experiments



Metric	Latest	Min	Max
train_loss	3.64 (step=0)	3.64 (step=0)	4.6 (step=0)
valid_loss	3.684 (step=0)	3.684 (step=0)	4.596 (step=0)

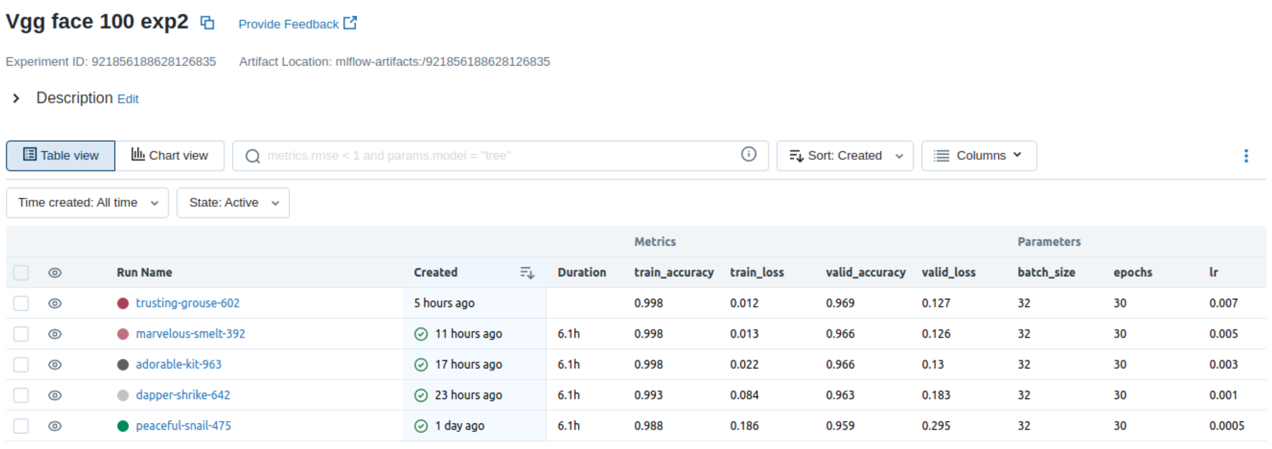
ResNet Train Loss Vs Validation Loss



Metric	Latest	Min	Max
train_accuracy	0.996 (step=0)	0.573 (step=0)	0.996 (step=0)
valid_accuracy	0.975 (step=0)	0.534 (step=0)	0.979 (step=0)

Resnet Train Accuracy Vs Validation Accuracy

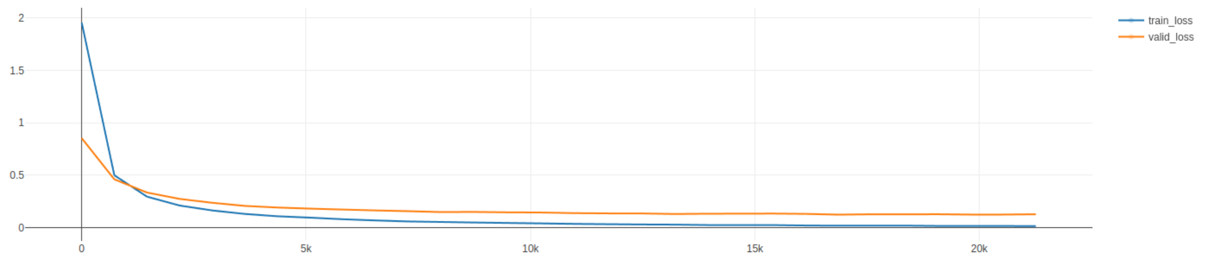
VGGFace

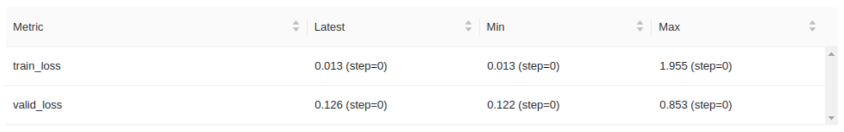

Vgg face 100 exp2

Experiment ID: 921856188628126835 Artifact Location: mlflow-artifacts:/921856188628126835

			Metrics				Parameters		
Run Name	Created	Duration	train_accuracy	train_loss	valid_accuracy	valid_loss	batch_size	epochs	lr
trusting-grouse-602	5 hours ago		0.998	0.012	0.969	0.127	32	30	0.007
marvelous-smelt-392	11 hours ago	6.1h	0.998	0.013	0.966	0.126	32	30	0.005
adorable-kit-963	17 hours ago	6.1h	0.998	0.022	0.966	0.13	32	30	0.003
dapper-shrike-642	23 hours ago	6.1h	0.993	0.084	0.963	0.183	32	30	0.001
peaceful-snail-475	1 day ago	6.1h	0.988	0.186	0.959	0.295	32	30	0.0005

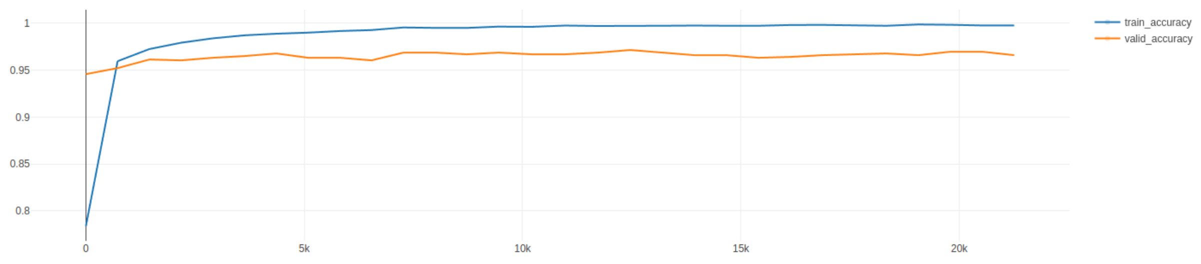
VGG Face mlflow experiments

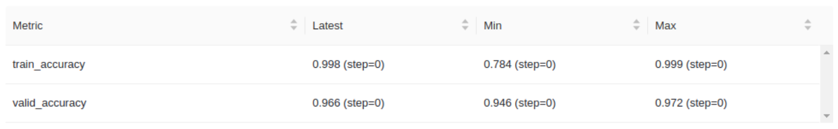




Metric	Latest	Min	Max
train_loss	0.013 (step=0)	0.013 (step=0)	1.955 (step=0)
valid_loss	0.126 (step=0)	0.122 (step=0)	0.853 (step=0)

VGG Face Train Loss Vs. Validation Loss





Metric	Latest	Min	Max
train_accuracy	0.998 (step=0)	0.784 (step=0)	0.999 (step=0)
valid_accuracy	0.966 (step=0)	0.946 (step=0)	0.972 (step=0)

VGG Face Train Accuracy Vs Validation Accuracy

Conclusions

- MTCNN gives higher accuracy than YOLO for Face Detection of multiple faces in an image, but YOLO works faster than MTCNN. So, for training, we are using YOLO as it contains only one image for recognition, and for the actual attendance marking mechanism we are using MTCNN to detect multiple faces.
- VGGFace & Inception ResNet V1 both are state-of-the-art methods for face recognition and give high accuracy, but VGGFace has more parameters than ResNet. So, the training of VGGFace is slower than ResNet.

Github Link:

Please find the github link below. The repository is not public. We have added you as collaborator.
https://github.com/kaushalmistry/FaceRecognitionBasedAttendanceSystem