Major ML Project

Face Recognition for Attendance System

CS550: Machine Learning

Group:

Name	Student ID
Abhishek Kumar	12140040
Aditya Tiwari	12140090
Arnav Gautam	12140280

Instructor: Prof. Gagan Raj Gupta

Department of Computer Science and Engineering
Indian Institute of Technology Bhilai
Durg, Bhilai, India

December 3, 2023

Contents

1 | Introduction

In an era marked by technological strides, this project focuses on Face Recognition for Attendance Systems, merging machine learning with the quest for efficient attendance tracking. Machine learning's capacity to discern patterns without explicit programming forms the backbone of this innovative approach. Traditional attendance methods, laden with inefficiencies and susceptibilities, make way for a secure and automated alternative.

Face recognition, driven by machine learning algorithms, not only promises accuracy but also mitigates the risks associated with manual or card-based systems. Beyond mere attendance automation, this project delves into the realm of heightened security, countering fraudulent activities through the unique digital signatures encoded in facial features. As we explore the fusion of machine learning and attendance tracking, the goal is to unravel the complexities involved in training models, selecting optimal algorithms, and adapting systems to real-world scenarios.

This project aligns with the 21st century's technological landscape, seeking to create a smart and efficient solution at the intersection of machine learning and attendance management.

2 | Objectives

The primary objectives of this Project are as follows:

- Identifying faces in input image.
- Classifying the identified faces as per the dataset.

3 | Requirements

The requirements that we used in the form of software / research papers / in order to perform this project:

1. Kaggle Sample Datasets
2. Google Colab, and its powerful GPUs
3. Papers with Code

4 | Methodology

Here is the step-by-step methodology of the project we did. You can add a bulleted list or flowchart (Figure 4.1).

- Data Loading and Preprocessing:
  - Upload a dataset using Google Colab.
  - Extract the dataset from a zip file.
  - Use ImageDataGenerator from TensorFlow to perform data augmentation and normalization.
  - Visualize the distribution of images across different classes in the dataset.
- Model Training (VGG16):
  - Use VGG16 architecture for face recognition.
  - Fine-tune the model on the provided dataset.
  - Save the trained VGG16 model for future use.
- Face Recognition using VGG16:
  - Use OpenCV, PIL, and the trained VGG16 model to perform face recognition on sample images.

- Display the predicted class label for each detected face.

- Data Loading for CNN Model:
  - Load the dataset for CNN training and testing.
- Feature Extraction (Facenet):
  - Use InceptionResNetV2 as a feature extraction model.
  - Extract embeddings for faces in both training and testing datasets.
- CNN Model Training:
  - Define a CNN model suitable for face recognition.
  - Train the CNN model on the normalized embeddings.
- Evaluate CNN Model:
  - Normalize input vectors and label encode targets.
  - Evaluate the CNN model on the test dataset.
  - Display accuracy scores and cross-validation results.
- Face Recognition using CNN:
  - Use OpenCV, PIL, and the trained CNN model for face recognition on sample images.
  - Display the predicted class label for each detected face.
- Load and Extract Faces from Images:
  - Load an image from a URL.
  - Use MTCNN to detect faces in the image.
  - Extract and resize the detected faces.
- Embedding Extraction for a New Face:
  - Use the Facenet model to extract embeddings for a new face.
  - Normalize the embedding and predict the class label using the SVM model.
- Display Predictions:
  - Display the input image with bounding boxes around detected faces.
  - Display the predicted class label for each detected face.

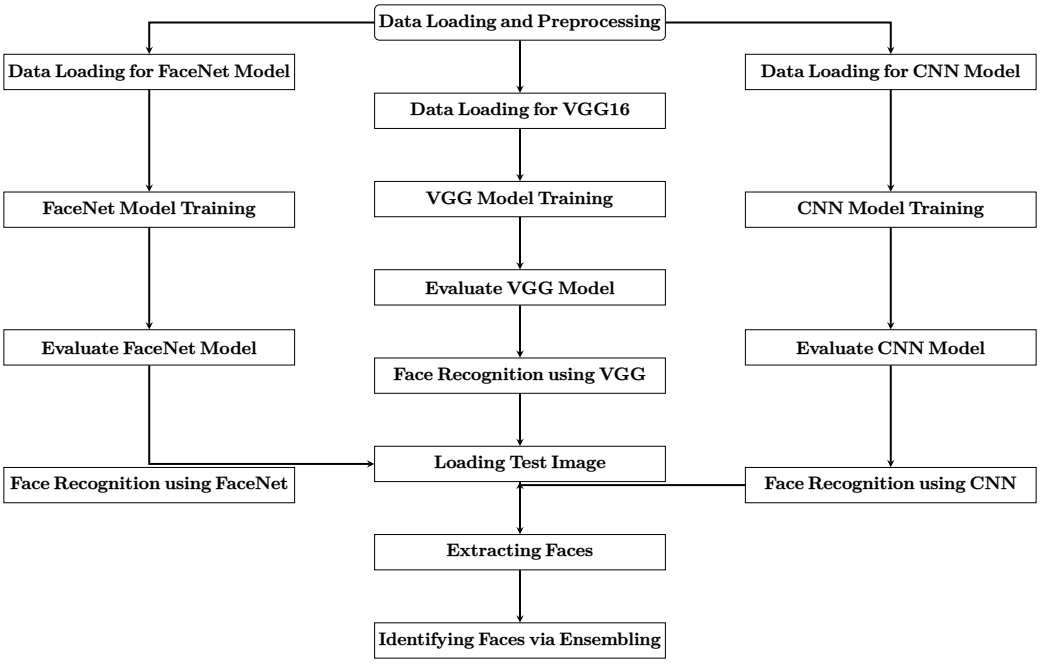

Figure 4.1: Methodology.

5 | Models

- **CNN:** Convolutional Neural Networks (CNNs) are a type of deep learning model designed for processing structured grid data, such as images. They use convolutional layers to automatically learn hierarchical patterns and features from the input data.

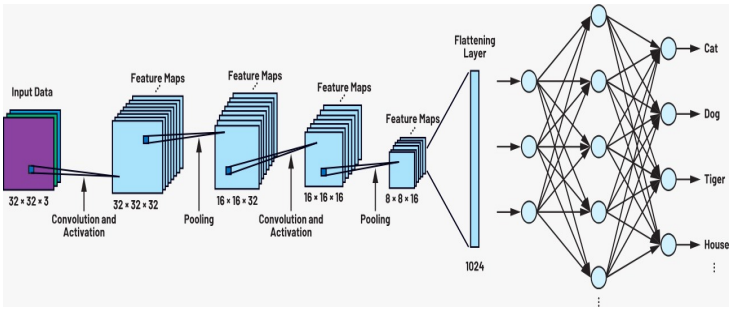

Figure 5.1: CNN Architecture

- **VGG16:** VGG16 is a pre-trained deep convolutional neural network architecture that consists of 16 weight layers, including convolutional and fully connected layers. It is known for its simplicity and effectiveness in image classification tasks.

- **FaceNet:** FaceNet is another pre-trained face recognition system that uses deep learning to map facial features into a high-dimensional space, enabling accurate face verification and identification. It employs a triplet loss function to optimize the embedding space for effective face clustering and recognition.

6 | Variance

Variance is the variability in the model prediction - how much the ML function can adjust depending on the given dataset.
We tried on one of the classes in our dataset, with the pictures that are not available in our dataset.

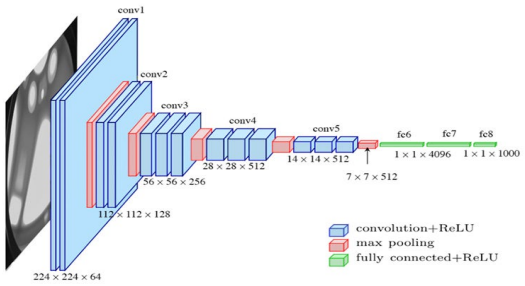

Figure 5.2: VGG16

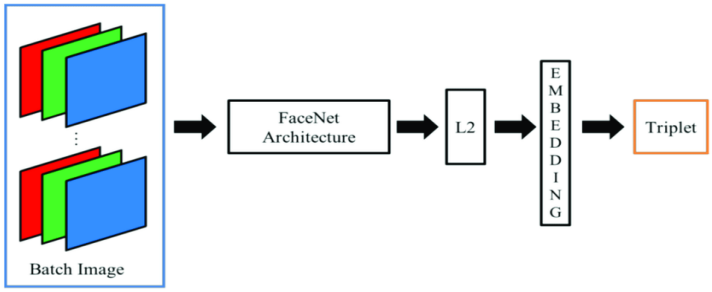

Figure 5.3: Facenet

We found the out of 10 images imported from the internet, on an average, we obtained 6 images as correct after testing the image and 4 incorrect, which shows that the variance in our model is 60%.

7 | Training Time

Table 7.1: Training Times

Model Name	Epochs	Training Time (sec)	Training Time(sec / epoch) / Epochs
FaceNet with SVM	-	150	-
CNN	20	120	6
VGG	20	120	6

(a) Facenet (b) Using VGG16 (c) Using CNN

Figure 7.1: Example of prediction result obtained from three models used

Table 7.2: Accuracies

Model Name	Train Accuracy	Validation Accuracy
FaceNet	100.00%	48.18%
CNN	95.00%	24.29%
VGG	90.91%	40.00%

8 | References

1. https://www.paperdigest.org/paper/?paper_id=doi.org_10.5244_c.29.41
2. https://www.semanticscholar.orgpaperFaceNet%3A-A-unified- embedding-for-face-recognition-Schroff-Kalenichenko/5aa26299435bdf7db874ef1640a6c3b5a4a2c394
3. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8