ATTENDANCE MONITORING USING FACE RECOGNITION

Department of Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology (Autonomous),
(Affiliated to Osmania University, Hyderabad)
Hyderabad, TELANGANA (INDIA) – 500 075
Month-20

-Submitted by
-S. SAI RAJ (160117733164)
-N. PRIYATAM (160117733160

CERTIFICATE

This is to certify that the project titled "**Attendance Monitoring Using Face Recognition**" is the bonafide work carried out by **S Sai Raj 160117733164, N Priyatam 160117733160** students of B.E.(CSE) of Chaitanya Bharathi Institute of Technology, Hyderabad, affiliated to Osmania University, Hyderabad, Telangana(India) during the academic year 2019-2020, submitted in partial fulfillment of the requirements for the **B.E.(CSE**) VI Semester Mini Projects and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Mentor(s)	**Head, CSE Dept.**
Sri B. Ramadasu,	Dr. M Swamy Das,
Assistant Professor	Professor

DECLARATION

We hereby declare that the project entitled “Attendance Monitoring Using Face Recognition” submitted for the B. E (CSE) Mini Projects is our original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

ACKNOWLEDGEMENT

We the authors of this project would like to express our immense gratitude to all the people involved in the successful execution of this project. We would like to take this opportunity to thank our mentor and guide Mr. Rama Dasu for his ever faithful and helping nature. Lastly, we would like to thank our classmates and peers, who supported and encouraged us throughout the project.

ABSTRACT

Authentication is a significant issue in system control in computer-based communication. Human face recognition is an important branch of biometric verification and has been widely used in many applications, such as video monitor system, human-computer interaction, and door control system and network security. This project describes a method for Student's Attendance System which will integrate with the face recognition technology using deep learning algorithms. The system will recognize the students present in the classroom and provide the list of present students for the lecture.

The primary technique used for the face detection is by using python inbuilt packages of OpenCV. Once the model is trained on different kinds of datasets, the project will help in identifying students present for the class. The front end will be based on an android application. The application uses SQLite database for establishing connection between web app and the model. The backend model mainly comprises of a convolutional neural network which extracts features and trains the model in recognizing those features. The inbuilt OpenCV uses haarcascade classifiers in identifying the faces present in the input image. The list of identified will be displayed as the end result.

INDEX

Chapter 1: Software Requirement Specification

1.1 Introduction

1.1.1 Problem Statement

1.1.1.1 Problem Definition:

The main aim of this project is to reduce the human intervention during the process of taking attendance, by introducing am automated attendance system using state of art face recognition technology.

1.1.1.2 Significance:

The project undertaken has immense significance and of paramount importance in an institutional ground. The project when implemented not only saves time otherwise spent on rollcall, it also reduces the human-prone errors. The system is useful in educational institution, front-line industries and any-where the concept of attendance applicable.

1.1.1.3 Objective:

The main objective of this project is the implementation of 'Attendance Monitoring using Face Recognition' which is the automatic update of attendance of students by face recognition. The lecturer takes a picture of the class and the students in the image are recognized and a list containing the recognized students is generated.

1.1.2 Scope

The main scope of this project is to equip the industry and institution with the sate-of- art technology in relation to attendance monitoring, with the proliferation of this system will not only reduce the latency of work, but also improve the efficiency of company as a whole.

1.2 General Description

1.2.1 Product Perspective

The mission of our project is to make the attendance monitoring system easier and efficient.

1.2.2 Product Functions

Login:

User should login into the application using her details (User ID & Password).

Class Credentials:

After successfully logging in, the user must provide the details of the subject taught and the section of the class being taught.

Image Input:

After submitting the details of the class, the application will prompt the user to input the image of all the students present for the class in a suitable format.

Result:

The application will open the spread sheet consisting of the details of all students present for the class.

1.2.3 User Characteristics

- User Friendly Environment
- User need not have any technical expertise to use the application
- A prior knowledge in the quality of image required for face recognition

1.2.4 General Constraints

- The Image format is restricted to JPEG or PNG
- The resolution of the camera should be at least 10MP.

1.3 Specific Requirements

1.3.1 Functional Requirements

1.3.1.1 Login:

The user needs to login with authenticate username and password to use the application.

1.3.1.1 a. Input:

The user enters his/her username and password.

1.3.1.1 b. Processing

User entered password and username would be subjected to authentication test.

1.3.1.1c. Output:

After verification, the page is directed to class credentials page.

1.3.1.2 Class credentials:

The user need to enter the details of subject and section taught.

1.3.1.2 a. Input:

The user needs to enter the subject and section of the class.

1.3.1.2 b. Processing:

User entered password and username would be subjected to authentication test.

1.3.1.2 c. Output:

The page is redirected to page for entering the image.

1.3.1.3 Image Input:

The user needs to input the image of students.

1.3.1.3 a. Input:

The user inputs the image of students in a jpeg or png format.

1.3.1.3 b. Processing:

Each student is identified in the picture using face recognition techniques and deep learning algorithms.

1.3.1.3.c. Output:

The list of students present and absent will be displayed

1.4 Non-Functional Requirements

1.4.1 Security Requirements

Every user will have separate login details, such that, they get logged into their account and take a picture for their respective period.

1.4.2 Safety Requirements

The system shall provide persistent storage where all the user details, list of students are stored.

1.4.3 Reliability Requirements

This application is more reliable. There will be no chance of malpractices.

1.4.4 Availability Requirements

Users can easily understand the process of the application and can access the application without any internet.

1.4.5 Maintainability Requirements

In case of any problem with the usage of application, a **help** button will be provided which answers basic questions regarding system crash, problems with recognition. Moreover, a complaint box will be provided to report any other problems for which immediate response will be given.

1.5 System Requirements

1.5.1 Software Requirements

1.5.1.1 Operating System:	Windows 10
1.5.1.2 Frameworks and IDE's:	Python Jupyter notebooks, Flask, Django.
1.5.1.3 Data Sets:	Image Datasets required for training the model.

1.5.2 Hardware Requirements

1.5.2.1 GPU:	Intel HD Graphics
1.5.2.2 CPU:	Intel i5 core
1.5.2.3 Camera:	Minimum 10megapixel
1.5.2.4 USB port:	1xUSB 2.0
1.5.2.5 RAM:	8GB

Chapter 2: Literature Survey

2.1 Introduction to Problem Domain

Attendance monitoring is the process which distinguishes the present from those who are absent in a fixed size group. Traditional methods of attendance monitoring have been keeping record by roll-call, signature taking and manual verification.

Facial Recognition is a technology capable of perceiving, identifying and recognizing a face from a digital image or video. The concept of Face Recognition is subdivided into Face Verification and Face Identification.

2.1.1 Face Verification:

It is the task of comparing a candidate face to another and verifying if it's a match. it's a one-to-one mapping.

2.1.2 Face Identification:

It is the task of extracting Facial Features and comparing the candidate Face with a database consisting of pre trained extracted facial features.

In this present project, the model of face recognition is trained on the images of students using deep learning technology, following which the facial features of the input image are extracted and compared against the pre-trained embeddings.

2.2 Existing Solution

2.2.1 Biometrics

Biometrics are physical or behavioral human characteristics to that can be used to digitally identify a person to grant access to systems, devices or data.

Examples of these biometric identifiers are fingerprints, facial patterns, voice or typing cadence. Each of these identifiers is considered unique to the individual, and they may be used in combination to ensure greater accuracy of identification

2.2.1.1 Fingerprint Authentication:

A form of biometric authentication, fingerprint authentication automatically compares a user's fingerprint to a stored fingerprint template in order to validate a user's identity. Because we are all born with unique fingerprints, fingerprint scans are an inherence factor or "something you are," making them impossible to guess and difficult to alter or fake.

A database of employee's or student's fingerprints is created by the authoritative figure of the institution. The employee or student can submit his fingerprint for verification of attendance.

2.2.1.2 Iris Scanner:

Iris recognition or iris scanning is the process of using visible and near-infrared light to take a high-contrast photograph of a person's iris. It is a form of biometric technology in the same category as face recognition and fingerprinting. Iris scanning measures the unique patterns in irises, the colored circles in people's eyes. Biometric iris recognition scanners work by illuminating the iris with invisible infrared light to pick up unique patterns that are not visible to the naked eye.

The digitalized Iris database is constructed by the authoritative figures of institution. The student or employee can have his/her iris scanned for marking of his presence in the institution

2.2.1.3 Facial Recognition:

A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape.

2.2.2 Logistics

Logistics are non-biometric implementation of IOT devices used to maintain, identify and verify the substances

2.2.2.1 Bar Code Scanning:

In this method, every student or employee is given a unique ID , unique within that particular organization. The unique ID is crypted into a bar code and displayed on card given to the user. The user can scan the barcode to identify and verify himself for attendance.

2.2.2.2 RFID tags:

Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects. An RFID tag consists of a tiny radio transponder; a radio receiver and transmitter. When triggered by an electromagnetic interrogation pulse from a nearby RFID reader device, the tag transmits digital data, usually an identifying inventory number, back to the reader.

2.3 Related Works

2.3.1 Face-Recognition Based Authentication

Unlike facial recognition which performs a 1:n match against a database of known faces, facial authentication is 1:1. The user is authenticating using their face as their credential to secure access to their online account. To authenticate, the user simply takes a selfie (which is converted into a 3D face map) that is then compared, one-to-one, with the stored biometric template. A proper match, based on an accuracy score, completes the secure authentication process in the background.

2.3.2 Face-Recognition Based Identification

While initially a form of computer application, it has seen wider uses in recent times on mobile platforms and in other forms of technology, such as robotics. It is typically used as access control in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. Although the accuracy of facial recognition system as a biometric technology is lower than iris recognition and fingerprint recognition, it is widely adopted due to its contactless and non-invasive process. Recently, it has also become popular as a commercial identification and marketing tool. Other applications include advanced human-computer interaction, video surveillance, automatic indexing of images, and video database, among others.

Chapter 3: Methodology and Design of Proposed System

3.1 Design of Proposed System

3.1.1 Block Diagram:

A **block diagram** is a **diagram** of a system in which the principal parts or functions are represented by **blocks** connected by lines that show the relationships of the **blocks**.

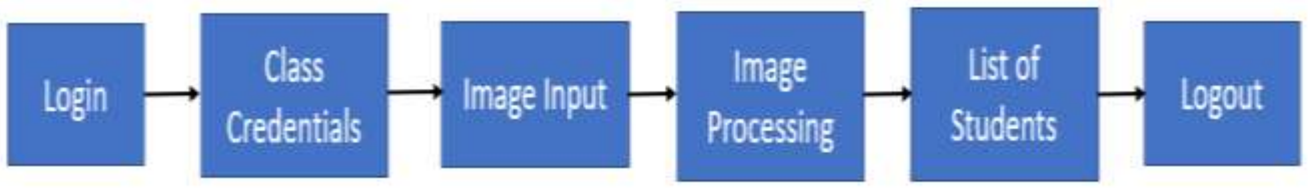


Figure 1: Block Diagram

The Block diagram above illustrates the basic design of the proposed system. The user is required to login, the user can then enter the class credentials of the class he wants to update of. The System processes the image and an excel sheet of students present and absent will be generated.

3.1.2 Data Flow Diagram:

DFD diagram shows the data flow between various processes in the system. DFD diagrams are drawn at different levels. Level 0 shows the main process in the system. Further levels describe the subprocesses and the development in the system.

Processes are represented by circles and data flow between them is represented by straight lines between the processes. External entities like input output are represented by rectangles. Data Store like Databases are represented by two parallel lines.

3.1.2.1 DFD level 0:

DFD level 0 is used for representation of the system as a whole without any implementation details. It provides a rough sketch of the system.

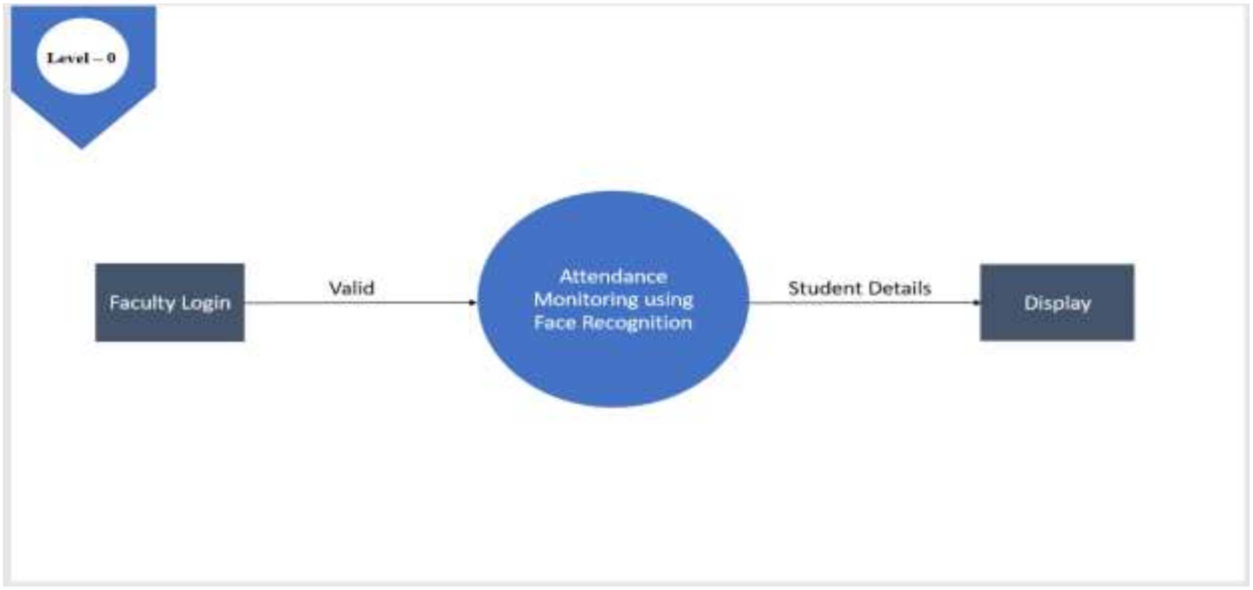


Figure 2: DFD level 0

3.2.1.2 DFD level 1:

DFD level 1 is used for representation of system with implementation details. It represents the flow of data in the system.

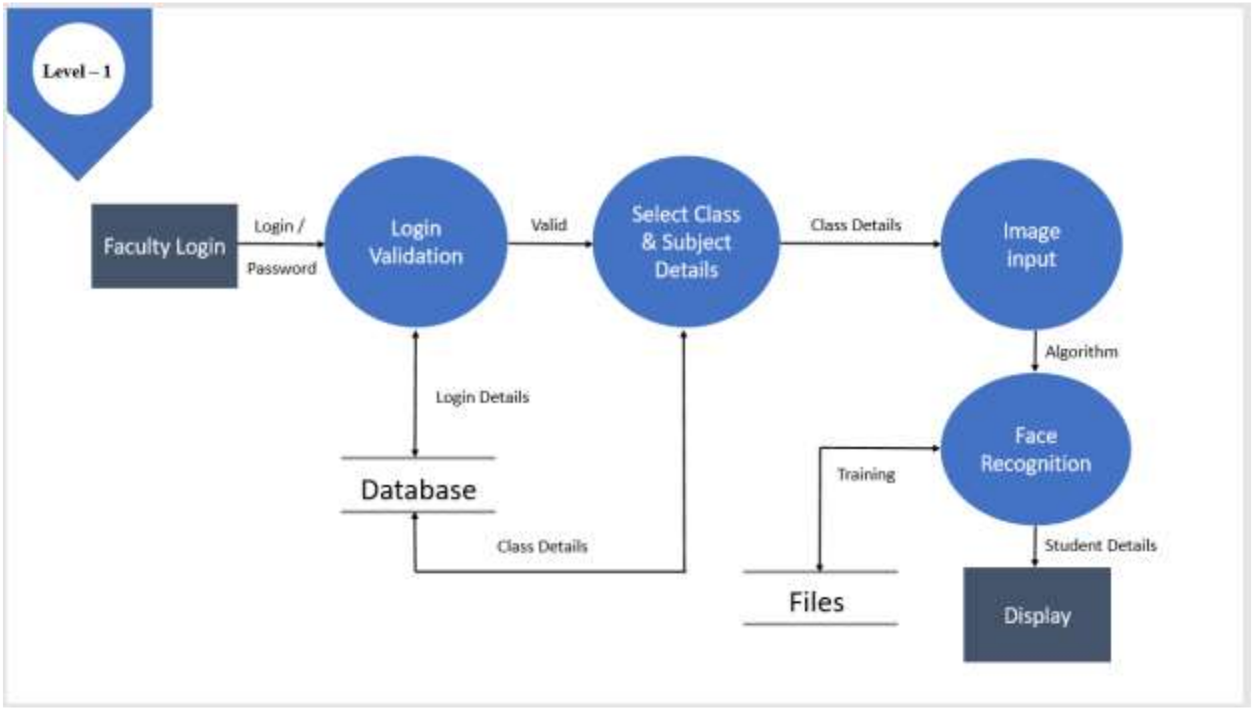


Figure 3: DFD level 1

3.2 Methodology

3.2.1 Convolutional Neural Networks

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

Architecture:

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that *convolve* with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point.

RELU Layer:

ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function It effectively removes negative values from an activation map by setting them to zero. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent and the sigmoid function ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

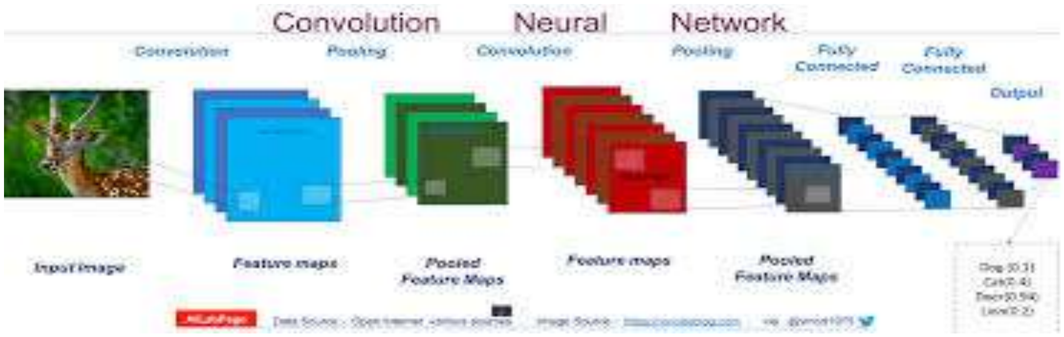


Figure 4: CNN

3.2.2 Support Vector Machine Classifier

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization classifier of the classifier.

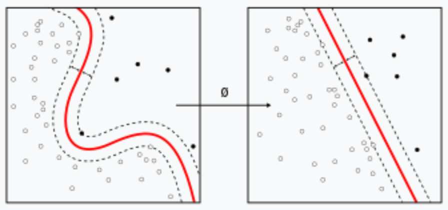


Figure 5: SVM

Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane.

.

Chapter 4: Implementation of Proposed System

4.1 Control Flow

Control flow diagrams dictate the flow of control through the system, they are basically following UML diagrams

4.1.1 Activity Diagram

Activity diagram shows various activities in the system and the order of the activities executed. An activity diagram starts with a start symbol and ends with end symbol. Activities are joined with arrowed lines. Diamonds are used to represent decision state. Each decision leads to a different activity. Fork is used to combine various activities. Activities are represented by round cornered rectangles fork is represented by black filled rectangle.

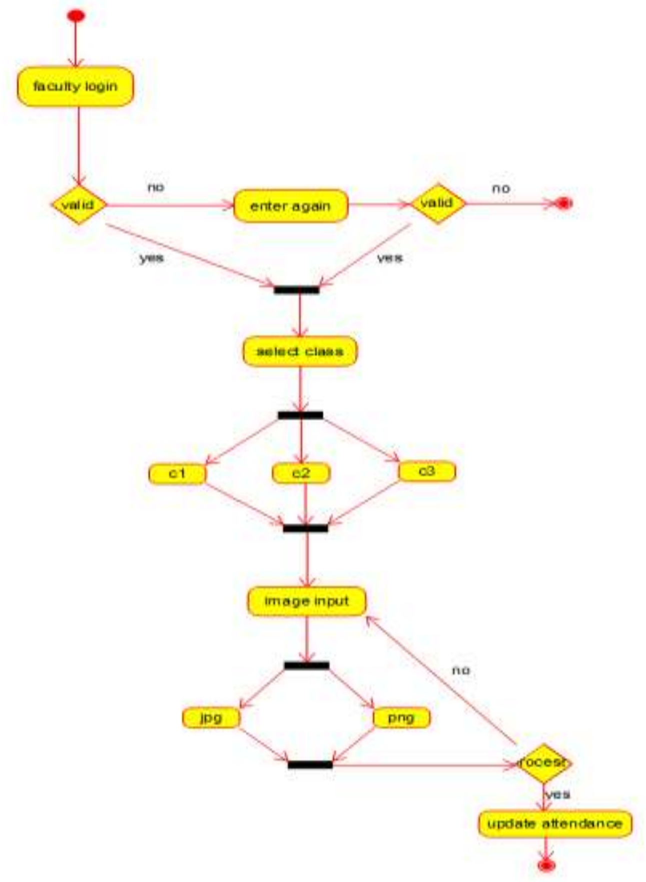


Figure 6: Activity Diagram

Various activities included in the system are as login, class credentials and image input. These activities are joined together through a logical flow as shown in the diagram

4.1.2 Use Case Diagram

Use case diagrams show the various actors who can interact with the system. Also the use cases that the system has. Use cases are like the features of the system. Use case diagram also shows which actor can interact with which use case. The relationship between an actor and a use case is an association and hence are joined by a straight line. Various use cases can be generalized.

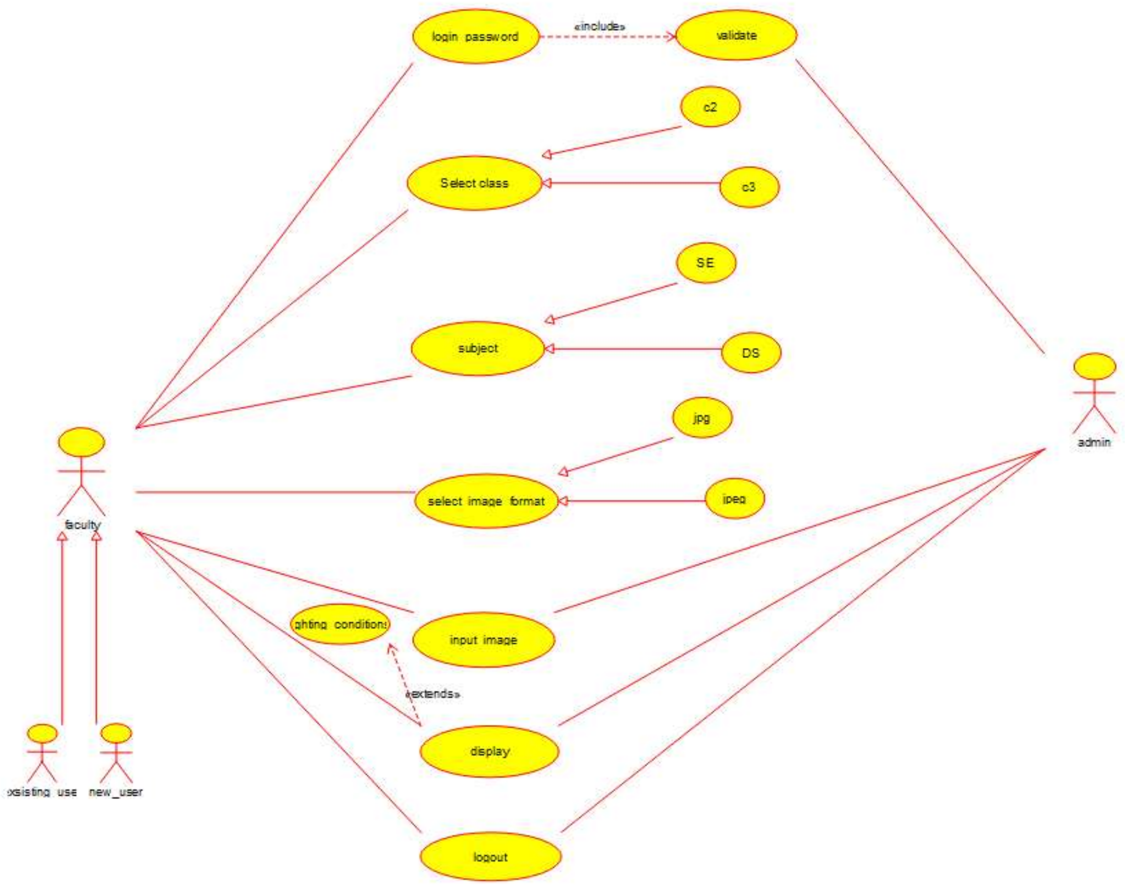


Figure 7: Use Case

Faculty and admin are two actors who are mainly involved in interaction with the system. the various use cases identified are login, class credentials, and image input. The admin oversees the image input for image processing and display.

4.2 FaceNet

Despite significant recent advances in the field of face recognition, implementing face verification and recognition efficiently at scale presents serious challenges to current approaches.

In this paper we present a system, called FaceNet, that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity.

Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.

Our method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. To train, we use triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method. The benefit of our approach is much greater representational efficiency: we achieve state-of-the-art face recognition performance using only 128-bytes per face.

Once this embedding has been produced, then the aforementioned tasks become straight-forward: face verification simply involves thresholding the distance between the two embeddings; recognition becomes a k-NN classification problem; and clustering can be achieved using off-the shelf techniques such as k-means or agglomerative clustering.

CNN using Stochastic Gradient Descent (SGD) with standard backprop and AdaGrad. In most experiments we start with a learning rate of 0.05 which we lower to finalize the model. The models are initialized from random, similar to, and trained on a CPU cluster for 1,000 to 2,000 hours. The decrease in the loss (and increase in accuracy) slows down drastically after 500h of training, but additional training can still significantly improve performance.

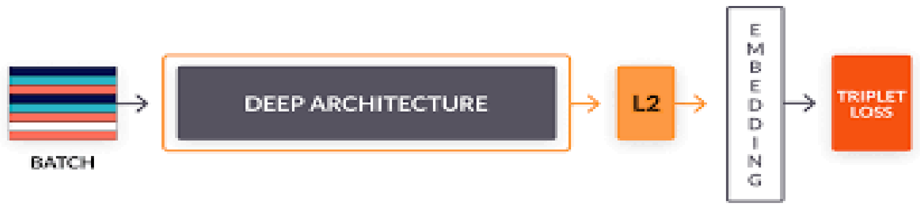


Figure 8: FaceNet Architecture

4.2.1 Triplet Loss

The embedding is represented by f(x) ∈ Rd It embeds an image x into a d-dimensional Euclidean space. Additionally, we constrain this embedding to live on the d-dimensional hypersphere, i.e. kf(x)k2 = 1. This loss is motivated in the context of nearest-neighbor classification. Here we want to ensure that an image x_i^a (anchor) of a specific person is closer to all other images x_i^p (positive) of the same person than it is to any image $x^n{}_i$ (negative) of any other person.

where α is a margin that is enforced between positive and negative pairs. T is the set of all possible triplets in the training set and has cardinality N.

$$Loss = \sum_{i=1}^{N} \left[\| f_i^a - f_i^p \|_2^2 - \| f_i^a - f_i^n \|_2^2 + \alpha \right]_+$$

In order to ensure fast convergence, it is crucial to select triplets that violate the triplet constraint in Eq. (1). This means that, given xa i, we want to select an xp i (hard positive) such that argmaxxp i kf(xa i)−f(xp i)k2 2 and similarly xn i (hard negative) such that argminxn i kf(xa i)−f(xn i)k2 2. It is infeasible to compute the argmin and argmax across the whole training set. Additionally, it might lead to poor training, as mislabeled and poorly imaged faces would dominate the hard positives and negatives.

To have a meaningful representation of the anchor positive distances, it needs to be ensured that a minimal number of exemplars of any one identity is present in each mini-batch. In our experiments we sample the training data such that around 40 faces are selected per-identity per-mini batch. Additionally, randomly sampled negative faces are added to each mini-batch.

Instead of picking the hardest positive, we use all anchor positive pairs in a mini-batch while still selecting the hard negatives. We don't have a side-by-side comparison of hard anchor-positive pairs versus all anchor-positive pairs within a mini-batch, but we found in practice that the all anchor positive method was more stable and converged slightly faster at the beginning of training.

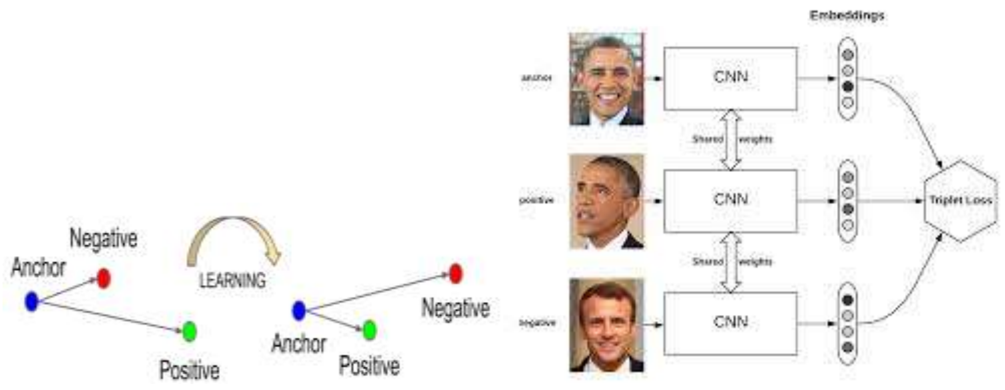


Figure 9: Triplet Loss

4.3 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program. The microframework Flask is based on the Pocoo projects Werkzeug and Jinja2.

Werkzeug:

Werkzeug is a utility library for the Python programming language, in other words a toolkit for Web Server Gateway Interface (WSGI) applications, and is licensed under a BSD License. Werkzeug can realize software objects for request, response, and utility functions. It can be used to build a custom software framework on top of it and supports Python 2.7 and 3.5 and later.

Jinja:

Jinja, also by Ronacher, is a template engine for the Python programming language and is l licensed under a BSD License. Similar to the Django web framework, it handles templates in a sandbox.

4.4 SQLite

SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, generally following PostgreSQL syntax. However, SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions, and will store the data as-is if such a conversion is not possible.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

SQLite implements most of the SQL-92 standard for SQL but it lacks some features. For example, it partially provides triggers, and it cannot write to views (however it provides INSTEAD OF triggers that provide this functionality). While it provides complex queries, it still has limited ALTER TABLE function, as it cannot modify or delete columns.

4.5 Training and Module Description

4.5.1 Loss Function:

The loss function used is Triplet loss function, as described in Chapter 3. The algorithm for triplet loss is as follows:

```
# Step 1: Compute the (encoding) distance between the anchor and the positive, you will need to sum
over axis=-1

   pos_dist = tf.reduce_sum(tf.square(tf.subtract(anchor, positive)), axis=-1)

   # Step 2: Compute the (encoding) distance between the anchor and the negative, you will need to sum
over axis=-1

   neg_dist = tf.reduce_sum(tf.square(tf.subtract(anchor, negative)), axis=-1)

   # Step 3: subtract the two previous distances and add alpha.

   basic_loss = tf.add(tf.subtract(pos_dist, neg_dist), alpha)

   # Step 4: Take the maximum of basic_loss and 0.0. Sum over the training examples.

   loss = tf.reduce_sum(tf.maximum(basic_loss, 0.0))
```

The Loss Function is defined as above and called each time while training, the loss function converged as follows after training:

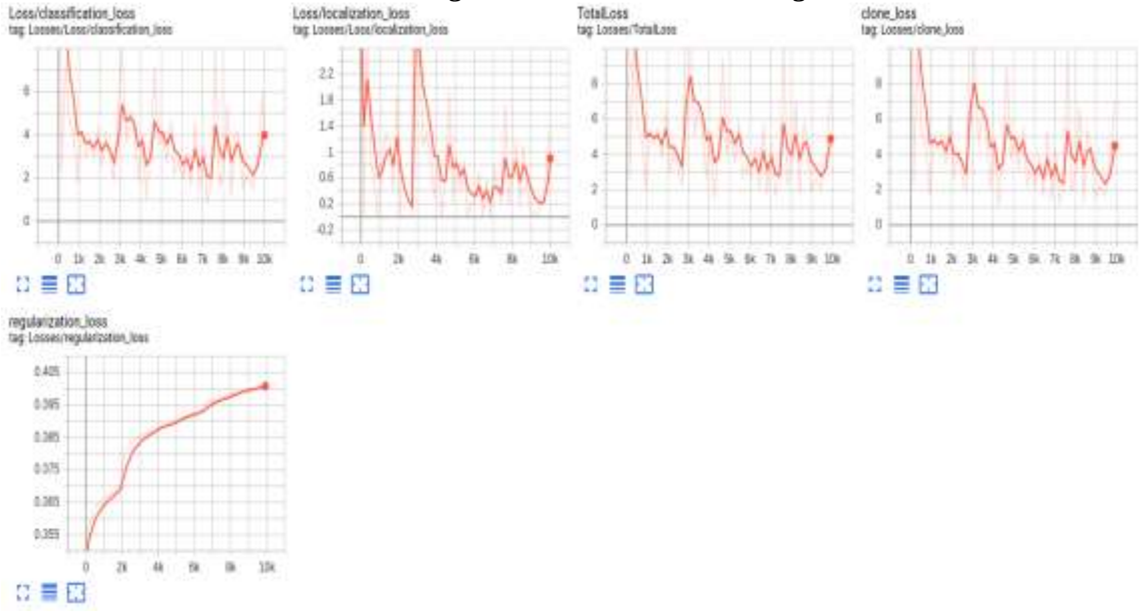


Figure 10: Output: Loss Output

4.5.2 Inception Module

The **Inception network** was an important milestone in the development of CNN classifiers. Prior to its **inception** (pun intended), most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance.

Its constant evolution lead to the creation of several versions of the network. image is the "naive" inception module. It performs **convolution** on an input, with **3 different sizes of filters** (1x1, 3x3, 5x5). Additionally, **max pooling** is also performed. The outputs are **concatenated** and sent to the next inception module.

The Algorithm for inception is as follows:

```
X_3x3 = Conv2D(96, (1, 1), data_format='channels_first', name ='inception_3a_3x3_conv1')(X)
   X_3x3 = BatchNormalization(axis=1, epsilon=0.00001, name = 'inception_3a_3x3_bn1')(X_3x3)
   X_3x3 = Activation('relu')(X_3x3)
   X_3x3 = ZeroPadding2D(padding=(1, 1), data_format='channels_first')(X_3x3)
   X_3x3 = Conv2D(128, (3, 3), data_format='channels_first', name='inception_3a_3x3_conv2')(X_3x3)
   X_3x3 = BatchNormalization(axis=1, epsilon=0.00001, name='inception_3a_3x3_bn2')(X_3x3)
   X_3x3 = Activation('relu')(X_3x3)
```

The Inception used is of version 3:

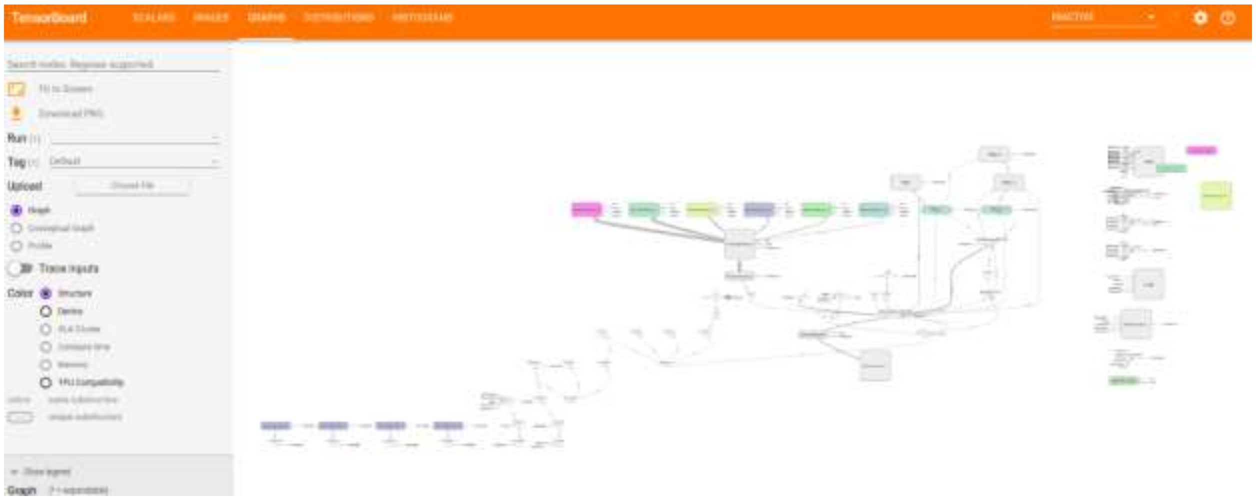


Figure 11: Output: Inception Hierarchy

4.5.3 Learning Module

The Learning Module Describes the Convolutional Pool built with activation Function 'ReLu' layer, with learning Rate (0.5) and Batch Normalization Techniques.

The Pseudocode is as follows:

```
def conv2d_bn(x, layer=None, cv1_out=None, cv1_filter=(1, 1),cv1_strides=(1, 1),cv2_out=None,cv2_filter=(3, 3),cv2_strides=(1, 1),padding=None):

    num = '' if cv2_out == None else '1'

    tensor = Conv2D(cv1_out, cv1_filter, strides=cv1_strides, data_format='channels_first', name=layer+'_conv'+num)(x)

    tensor = BatchNormalization(axis=1, epsilon=0.00001, name=layer+'_bn'+num)(tensor)

    tensor = Activation('relu')(tensor)

    if padding == None:return tensor

    tensor = ZeroPadding2D(padding=padding, data_format='channels_first')(tensor)

    if cv2_out == None:return tensor

 tensor = Conv2D(cv2_out, cv2_filter, strides=cv2_strides, data_format='channels_first', name=layer+'_conv'+'2')(tensor tensor = BatchNormalization(axis=1, epsilon=0.00001, name=layer+'_bn'+'2')(tensor) tensor = Activation('relu')(tensor)
```

As the training proceeded, the Learning Rate has changed as follows:

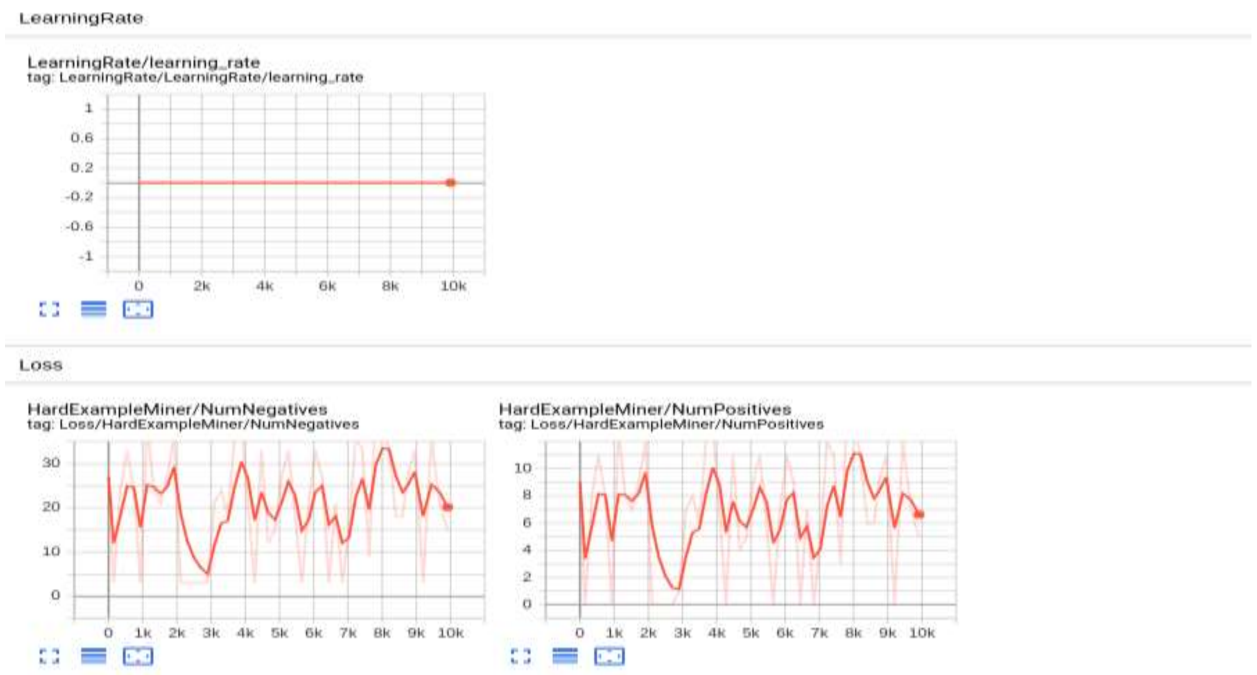


Figure 12: Output: Learning Rate

4.6 Data Set Description

This Project has required the Authors to create our own custom dataset. The Dataset built is an Images Folder. The Images Folder consists of Individual Folders containing 15- pictures if the people that have to identified and recognized by the System. The Individual Folders are named according to the names stored in Database. The outline of Dataset is as follows,

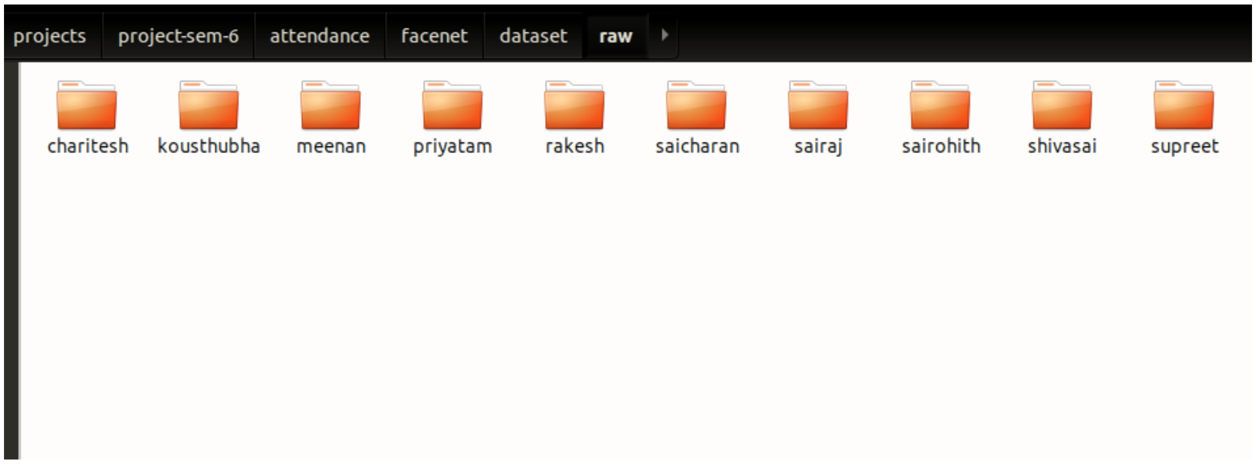


Figure 13: Data Set

4.7 Testing Process

The System built can be validated for accuracy and correctness by testing the model on the images that have not been trained upon. This can include the pictures of people the model has not been trained on.

4.8.1 Test-Case Design

Test- case Design entails the process of constructing test where the system is likely to fail. The proposed system can be tested for validation by giving the input, pictures of people the model has not been trained on, the pictures not containing human faces, pictures of people without complete facial features, low pixel quality pictures, etc.

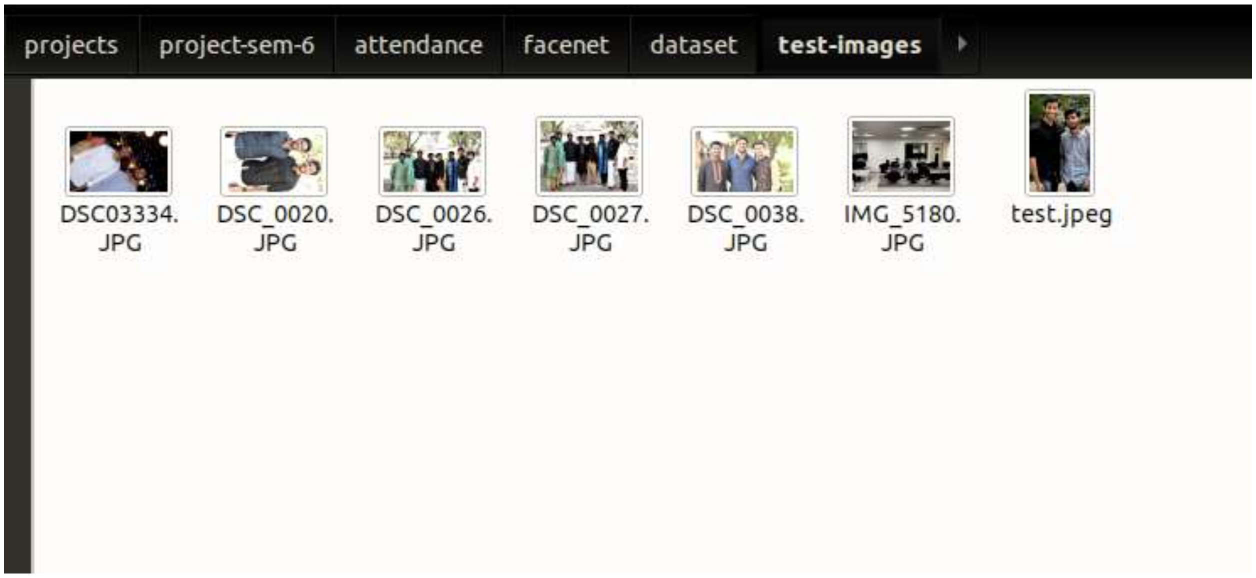


Figure 14: Test Set

Chapter 5: Results and Output

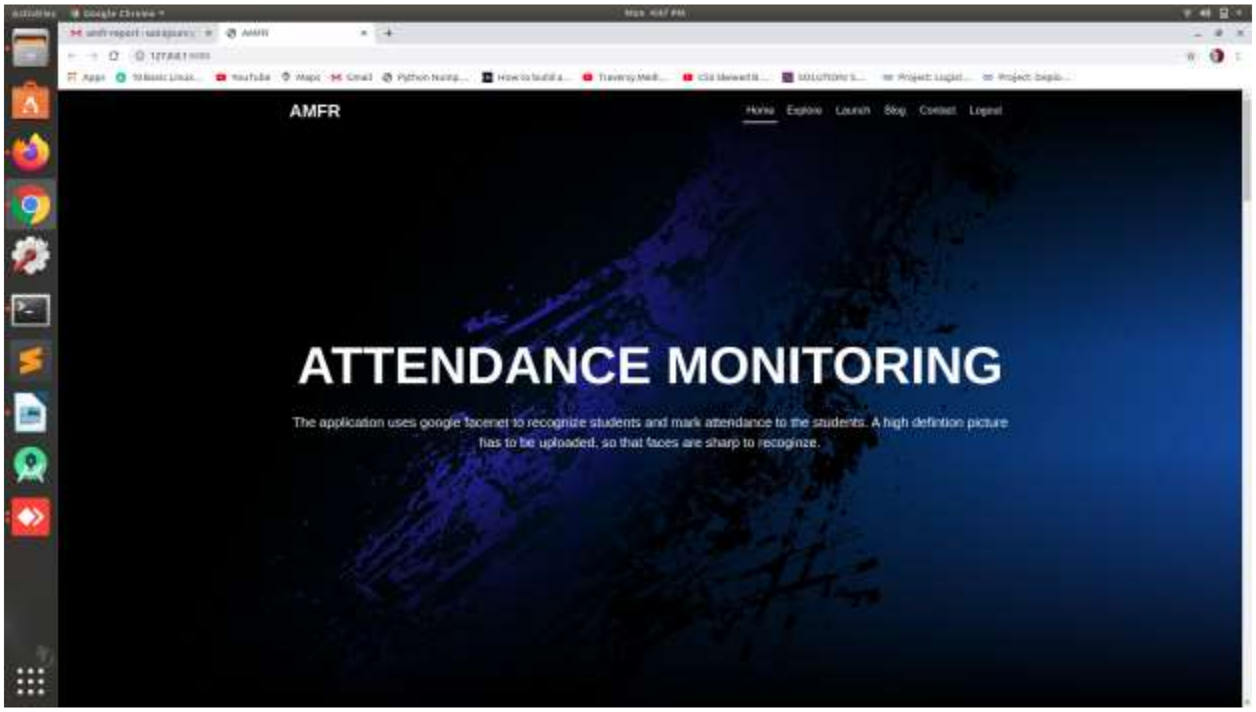

AMFR
Home
Explore
Launch
Blog
Contact
Logout
ATTENDANCE MONITORING
The application uses google facenet to recognize students and mark attendance to the students. A high definition picture has to be uploaded, so that faces are sharp to recognize.

Figure 15: Output: Title Page

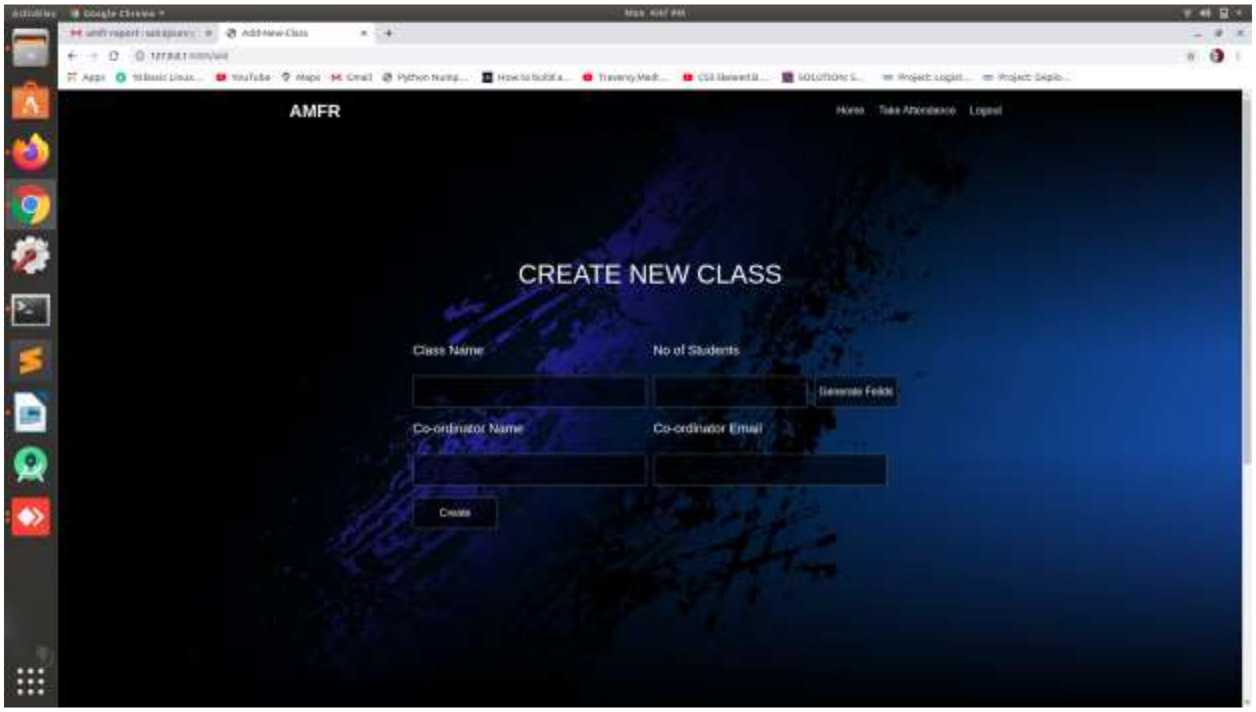


Figure 16: Output: Class Credentials

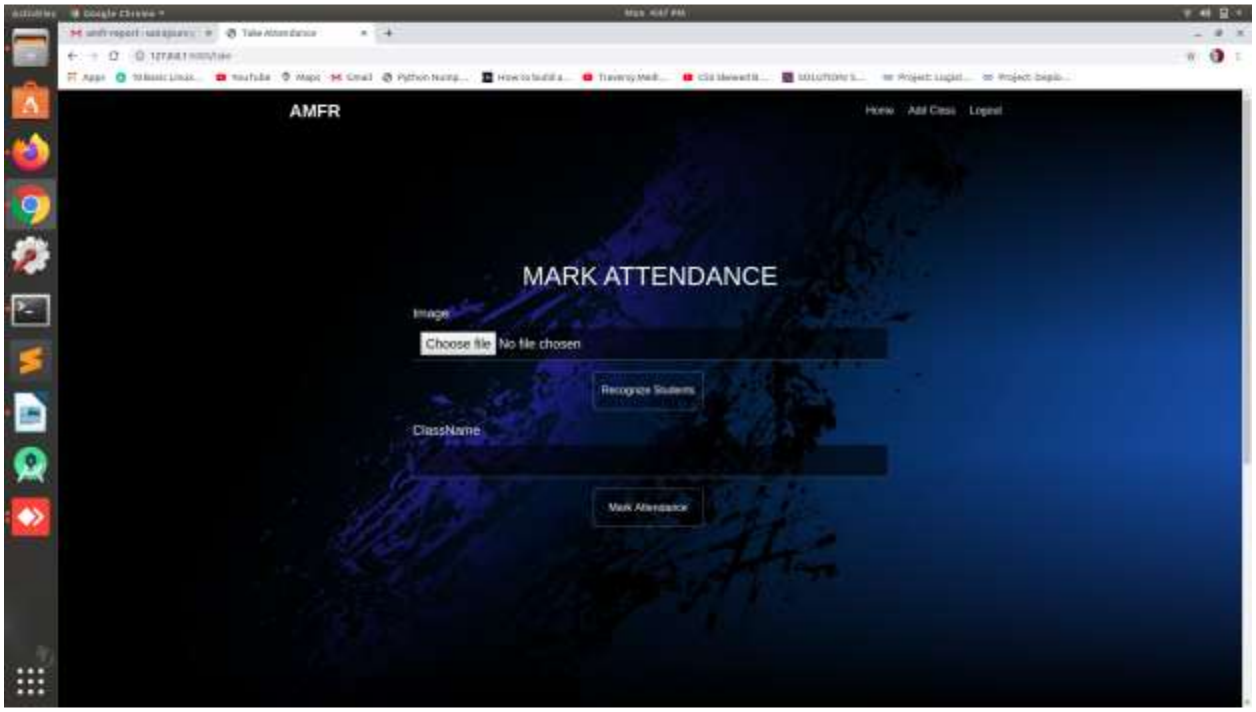


Figure 17: Output: Marking Attendance

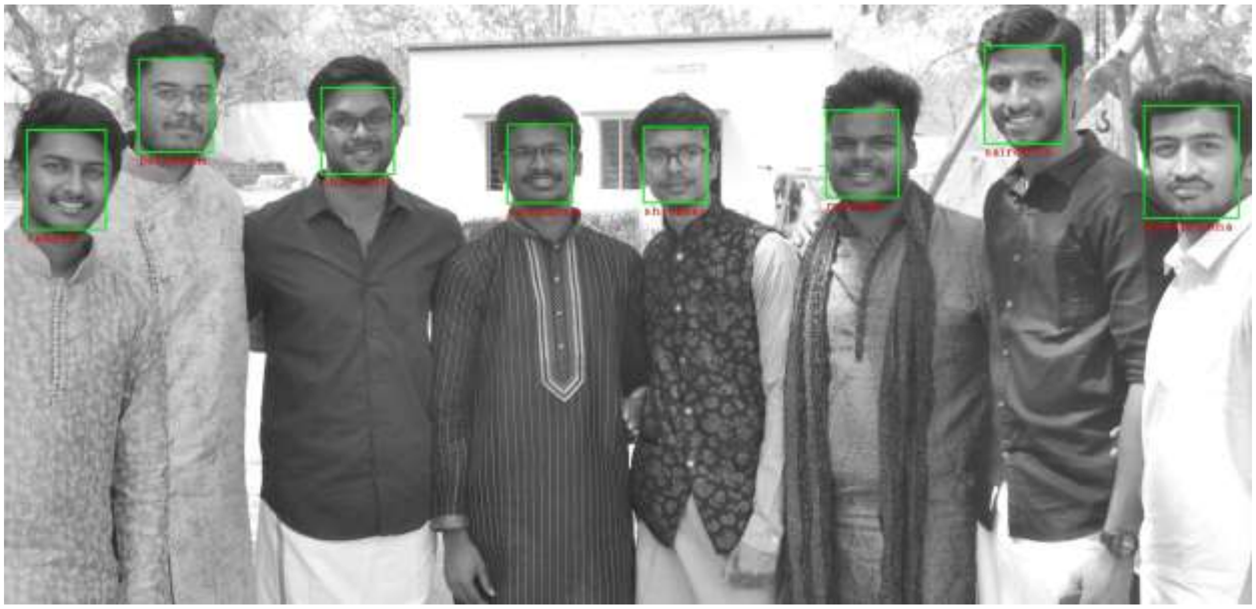
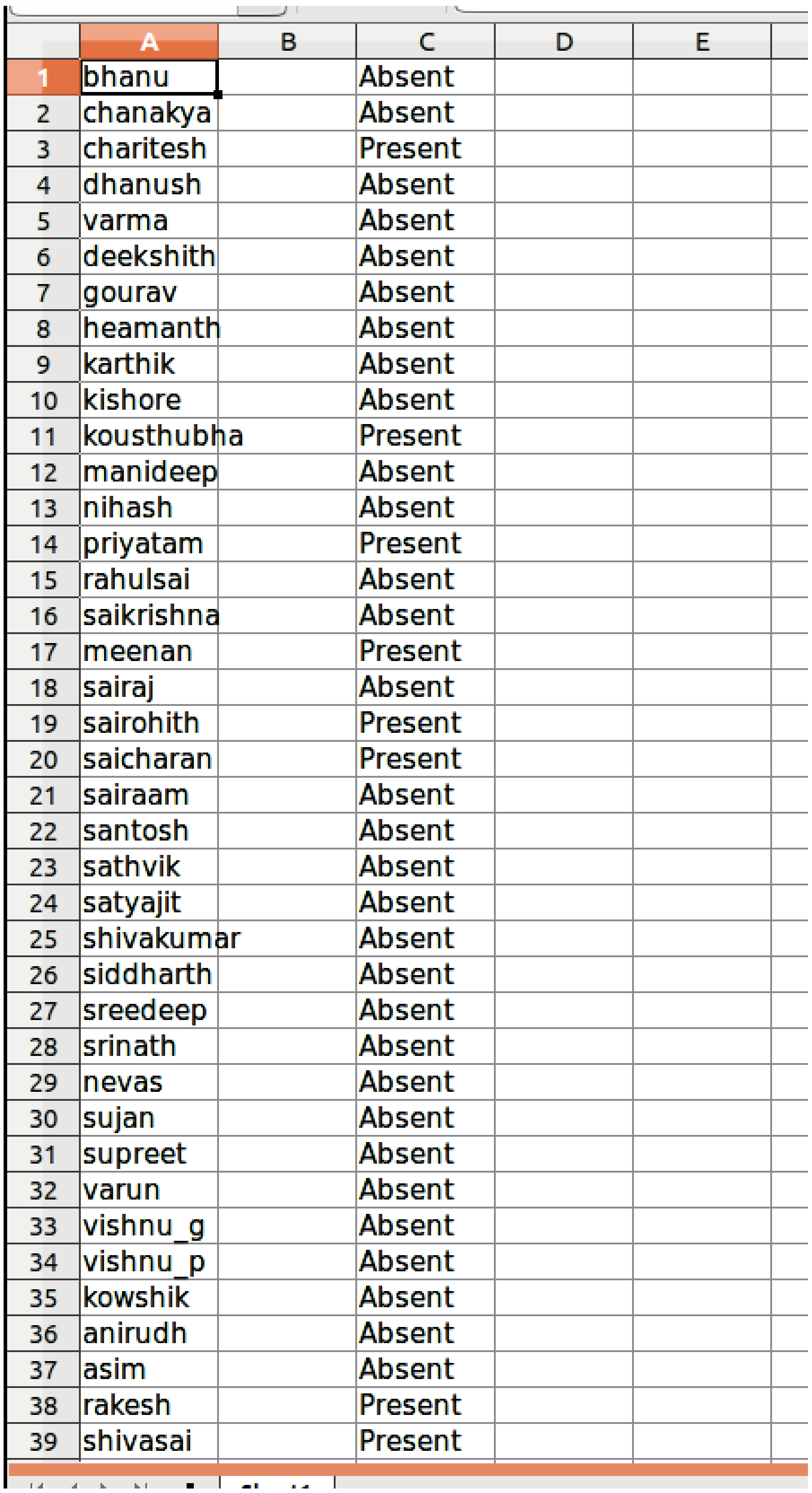
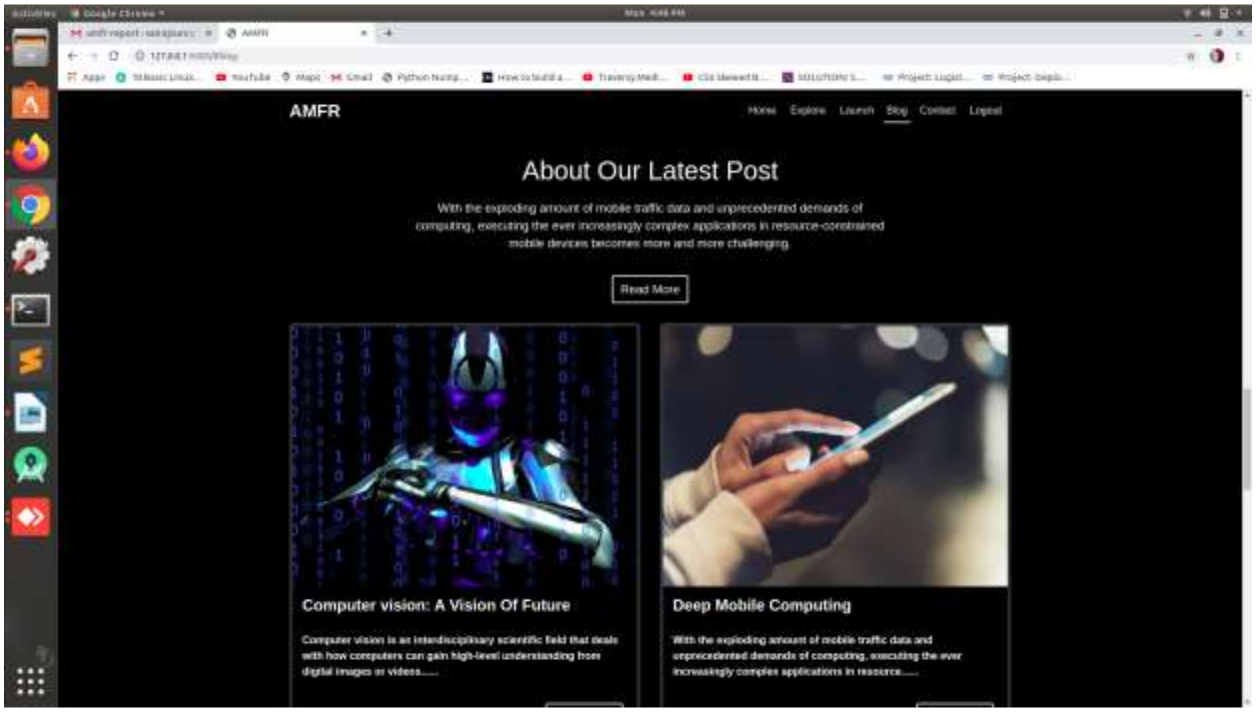


Figure 18: Output: Faces Recognized



	A	B	C	D	E
1	bhanu		Absent		
2	chanakya		Absent		
3	charitesh		Present		
4	dhanush		Absent		
5	varma		Absent		
6	deekshith		Absent		
7	gourav		Absent		
8	heamanth		Absent		
9	karthik		Absent		
10	kishore		Absent		
11	kousthubha		Present		
12	manideep		Absent		
13	nihash		Absent		
14	priyatam		Present		
15	rahulsai		Absent		
16	saikrishna		Absent		
17	meenan		Present		
18	sairaj		Absent		
19	sairohith		Present		
20	saicharan		Present		
21	sairaam		Absent		
22	santosh		Absent		
23	sathvik		Absent		
24	satyajit		Absent		
25	shivakumar		Absent		
26	siddharth		Absent		
27	sreedeep		Absent		
28	srinath		Absent		
29	nevas		Absent		
30	sujan		Absent		
31	supreet		Absent		
32	varun		Absent		
33	vishnu_g		Absent		
34	vishnu_p		Absent		
35	kowshik		Absent		
36	anirudh		Absent		
37	asim		Absent		
38	rakesh		Present		
39	shivasai		Present		

Figure 19: Output: Attendance Report



About Our Latest Post

With the exploding amount of mobile traffic data and unprecedented demands of computing, executing the ever increasingly complex applications in resource-constrained mobile devices becomes more and more challenging.

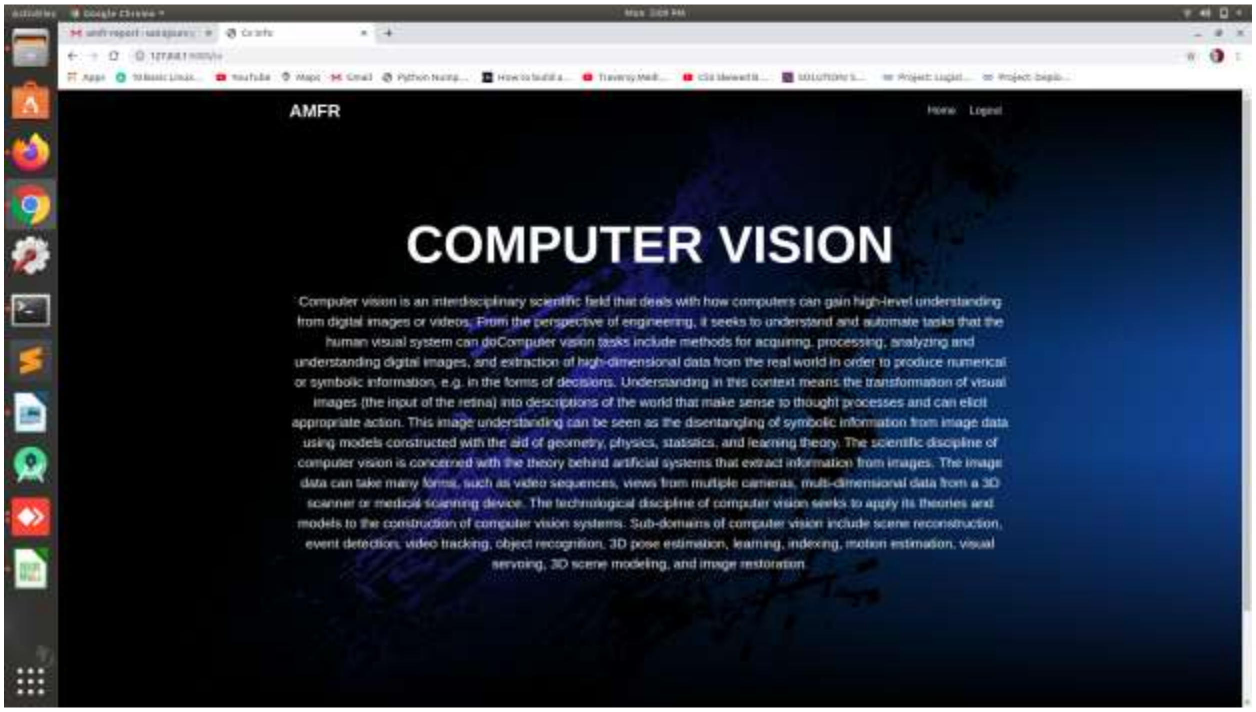
Computer vision: A Vision Of Future

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos......

Deep Mobile Computing

With the exploding amount of mobile traffic data and unprecedented demands of computing, executing the ever increasingly complex applications in resource......

Figure 20: Output: Blog



COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can doComputer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that make sense to thought processes and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory. The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multi-dimensional data from a 3D scanner or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems. Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual servoing, 3D scene modeling, and image restoration.

Figure 21: Output: Blog Content

Conclusion

We author hereby conclude that the proposed system attendance monitoring using face recognition has been successfully implemented with utmost accuracy and precision. The project has scope for further improvements such as real time database integration, android deployment etc. We author thank everyone involved in this project.

References

1. https://ieeexplore.ieee.org/document/7298682

2. https://medium.com/@teyou21/setup-tensorflow-for-object-detection-on-ubuntu-16-04-e2485b52e32a

3. https://medium.com/@teyou21/training-your-object-detection-model-on-tensorflow-part-2-e9e12714bdf

4. https://medium.com/@teyou21/convert-a-tensorflow-frozen-graph-to-a-tflite-file-part-3-1ccdb3874c4a

Appendix

run.py:

```
from attendance import app
from attendance import db
if __name__ == '__main__':
        db.create_all()
        app.run(debug=True)
```

__init__.py:

```
from flask_sqlalchemy import SQLAlchemy
from flask import Flask
from flask_bcrypt import Bcrypt
from flask_login import LoginManager

app = Flask(__name__)
app.config['SECRET_KEY'] = '59d4e34a8c7c47b529b7e73b461982c4'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'
login_manager.login_message_category = 'info'

from attendance import main
```

main.py:

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

from flask import Flask
from flask import render_template,url_for,flash,redirect,request,jsonify,abort,make_response
from attendance import app, db, bcrypt
from attendance.models import User,Add
from flask_login import login_user, current_user, logout_user, login_required
from attendance.forms import RegistrationForm, LoginForm, AddForm, EditForm

import os
import pickle
import sys
import time
import cv2
import numpy as np
import tensorflow as tf
from scipy import misc
import attendance.facenet.src.facenet as facenet
from attendance.facenet.src.align import detect_face
from keras.models import load_model
from flask_httpauth import HTTPBasicAuth
import sqlite3
import xlsxwriter
import datetime
import requests
from PIL import Image
from werkzeug.utils import secure_filename

auth = HTTPBasicAuth()
names = []

@app.route('/')
def index():
                return render_template('index.html',title='homepage')
```

```
@app.route('/login', methods=['POST','GET'])
def login():
                if current_user.is_authenticated:
                    return redirect(url_for('index'))
                form = LoginForm()
                if form.validate_on_submit():
                    user = User.query.filter_by(email=form.email.data).first()
                    if user and bcrypt.check_password_hash(user.password, form.password.data):
                        login_user(user, remember=form.remember.data)
                        flash('You have been logged in!','success')
                        next_page = request.args.get('next')
                        if next_page:
                            return redirect(next_page)
                        else:
                            return redirect(url_for('index'))
                    else:
                        flash('Login Unsuccessful. Please check Email and Password','danger')
                return render_template('login.html',title='Login',form=form)


@app.route("/register", methods=['GET','POST'])
def register():
                if current_user.is_authenticated:
                    return redirect(url_for('index'))
                form = RegistrationForm()
                if form.validate_on_submit():
                    hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
                    user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
                    db.session.add(user)
                    db.session.commit()
                    flash('Your account has been created! You are now able to Log In', 'success')
                    return redirect(url_for('login'))
```

```
        return render_template('register.html',title='Register',form=form)


@app.route("/logout")
def logout():
        logout_user()
        return redirect(url_for('index'))


@app.route("/add", methods=['GET','POST'])
@login_required
def add_class():
        cn=request.form.get('classname')
        n=request.form.get('noofstu')
        coorn=request.form.get('coorname')
        co_email=request.form.get('cooremail')
        fn_list=request.form.getlist('namefields[]')
        fp_list=request.form.getlist('phonefields[]')
        fr_list=request.form.getlist('rollfields[]')
        if fn_list!=None and fp_list!=None and fr_list!=None and n!=None:
            print("hi",fn_list,fp_list,fr_list,n)
            for i in range(int(n)):
                new=Add(classname=cn, coordinator=coorn, co_email=co_email,
stuname=fn_list[i], regno=fr_list[i], mobileno=fp_list[i])
                db.session.add(new)
            db.session.commit()
            flash('A new class has been created!','success')
            return redirect(url_for('index'))
        return render_template('add_class.html',title='Add New Class')


@app.route("/take")
@login_required
def take_attendance():
        return render_template('take_attendance.html',title="Take Attendance")
```

```
@app.route("/recognition")
def recognition():
                    return render_template('recog.html',title="Recognized students")


@app.route("/face_recog", methods=['GET','POST'])
def face_recog():
                    global names
                    image = request.files['image']
                    nom_image = secure_filename(image.filename)
                    image = Image.open(image)
                    image.save('/home/sai/sai/projects/project-sem-6/attendance/facenet/dataset/test-
images/'+nom_image)
                    img_name=str(nom_image)
                    img_path="attendance/facenet/dataset/test-images/"+img_name
                    modeldir = "attendance/facenet/src/20180402-114759/"
                    classifier_filename = "attendance/facenet/src/20180402-114759/classifier.pkl"
                    npy=""
                    train_img="attendance/facenet/dataset/raw"


                    with tf.Graph().as_default():
                        gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.6)
                        sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options,
log_device_placement=False))
                        with sess.as_default():
                                pnet, rnet, onet = detect_face.create_mtcnn(sess, npy)


                                minsize = 20  # minimum size of face
                                threshold = [0.6, 0.7, 0.7]  # three steps's threshold
                                factor = 0.709  # scale factor
                                margin = 32
                                frame_interval = 3
```

```
batch_size = 1000
image_size = 160
input_image_size = 160

HumanNames = os.listdir(train_img)
HumanNames.sort()

print('Loading feature extraction model')
facenet.load_model(modeldir)

images_placeholder = tf.get_default_graph().get_tensor_by_name("input:0")
embeddings = tf.get_default_graph().get_tensor_by_name("embeddings:0")
phase_train_placeholder =
tf.get_default_graph().get_tensor_by_name("phase_train:0")
embedding_size = embeddings.get_shape()[1]

classifier_filename_exp = os.path.expanduser(classifier_filename)
with open(classifier_filename_exp, 'rb') as infile:
        (model, class_names) = pickle.load(infile)
# video_capture = cv2.VideoCapture("akshay_mov.mp4")
c = 0

print('Start Recognition!')
prevTime = 0
# ret, frame = video_capture.read()
frame = cv2.imread(img_path,0)
frame = cv2.resize(frame, (0,0), fx=0.5, fy=0.5)    #resize frame (optional)
curTime = time.time()+1    # calc fps
timeF = frame_interval
if (c % timeF == 0):
        find_results = []
        if frame.ndim == 2:
```

```
                    frame = facenet.to_rgb(frame)
                frame = frame[:, :, 0:3]
                bounding_boxes, _ = detect_face.detect_face(frame, minsize, pnet,
rnet, onet, threshold, factor)
                nrof_faces = bounding_boxes.shape[0]
                print('Face Detected: %d' % nrof_faces)
                if nrof_faces > 0:
                    det = bounding_boxes[:, 0:4]
                    img_size = np.asarray(frame.shape)[0:2]
                    cropped = []
                    scaled = []
                    scaled_reshape = []
                    bb = np.zeros((nrof_faces,4), dtype=np.int32)
                for i in range(nrof_faces):
                    emb_array = np.zeros((1, embedding_size))
                    bb[i][0] = det[i][0]
                    bb[i][1] = det[i][1]
                    bb[i][2] = det[i][2]
                    bb[i][3] = det[i][3]
                    #inner exception
                    if bb[i][0] <= 0 or bb[i][1] <= 0 or bb[i][2] >= len(frame[0])
or bb[i][3] >= len(frame):
                        print('face is too close')
                        break
                    cropped.append(frame[bb[i][1]:bb[i][3], bb[i][0]:bb[i][2], :])
                    cropped[i] = facenet.flip(cropped[i], False)
                    scaled.append(misc.imresize(cropped[i], (image_size,
image_size), interp='bilinear'))
                    scaled[i] = cv2.resize(scaled[i],
(input_image_size,input_image_size),

interpolation=cv2.INTER_CUBIC)
                    scaled[i] = facenet.prewhiten(scaled[i])
```

```
                                                            scaled_reshape.append(scaled[i].reshape(-
1,input_image_size,input_image_size,3))
                                                            feed_dict = {images_placeholder: scaled_reshape[i],
phase_train_placeholder: False}
                                                            emb_array[0, :] = sess.run(embeddings, feed_dict=feed_dict)
                                                            predictions = model.predict_proba(emb_array)
                                                            #print(predictions)
                                                            best_class_indices = np.argmax(predictions, axis=1)
                                                            # no print(best_class_indices)
                                                            best_class_probabilities =
predictions[np.arange(len(best_class_indices)), best_class_indices]
                                                            #print(best_class_probabilities)
                                                            cv2.rectangle(frame, (bb[i][0], bb[i][1]), (bb[i][2], bb[i][3]),
(0, 255, 0), 2)    #boxing face
                                                            #plot result idx under box
                                                            text_x = bb[i][0]
                                                            text_y = bb[i][3] + 20
                                                            #print(HumanNames,best_class_indices)
                                                            #print('Result Indices: ', best_class_indices[0])
                                                            print(HumanNames[best_class_indices[0]])
                                                            names.append(HumanNames[best_class_indices[0]])
                                                            for H_i in HumanNames:
                                                                    if HumanNames[best_class_indices[0]] == H_i:
                                                                            result_names =
HumanNames[best_class_indices[0]]
                                                                            cv2.putText(frame, result_names, (text_x,
text_y), cv2.FONT_HERSHEY_COMPLEX_SMALL,1, (0, 0, 255), thickness=1, lineType=1)
                                            else:
                                                    print('Unable to align')

                        # reg_no = c.execute("SELECT regno FROM 'add'")
                        # for i, row in enumerate(reg_no):
                        #       for j, value in enumerate(row):
                        #               worksheet.write(i,j+1,value)
```

```
			cv2.imwrite('/home/sai/sai/projects/project-sem-6/attendance/output.jpg',frame)
			#cv2.imwrite('output/'+img_path.split('/')[-1],frame)
			return render_template('take_attendance.html',title="Take Attendance")

@app.route("/mark", methods=['GET','POST'])
def mark():
			global names
			classnm=request.form.get('classid')
			print(classnm)
			workbook = xlsxwriter.Workbook('/home/sai/sai/projects/project-sem-
6/attendance/Reports/Report_for_'+ datetime.datetime.now().strftime("%Y_%m_%d-%H")+'.xlsx')
			worksheet = workbook.add_worksheet()
			conn = sqlite3.connect('/home/sai/sai/projects/project-sem-6/attendance/site.db')
			c = conn.cursor()
			students = c.execute("select stuname from 'add' where classname='%s'" % classnm)
			for i, row in enumerate(students):
			    for j, value in enumerate(row):
				    worksheet.write_string(i,j+2,'Absent')
				    for name in names:
					    if name == value:
						    worksheet.write_string(i,j+2,'Present')
				    worksheet.write_string(i,j, str(value))
			workbook.close()
			names=[]
			flash('The students report generated successfully!','success')
			return render_template('take_attendance.html',title="Take Attendance")

@app.route("/sms", methods=['GET','POST'])
def sms():
			return render_template('take.html',title="Take Attendance")

@app.route("/cv")
```

```
def cv():
        return render_template('cv.html',title="Cv")


@app.route("/dmc")
def dmc():
        return render_template('dmc.html',title="deep mobile computing")


@app.route("/flsk")
def flsk():
        return render_template('flsk.html',title="Flask Info")


@app.route("/sqlyt")
def sqlyt():
        return render_template('sqlyt.html',title="SQL Info")
```

facenet.py:

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function


import os
from subprocess import Popen, PIPE
import tensorflow as tf
import numpy as np
from scipy import misc
from sklearn.model_selection import KFold
from scipy import interpolate
from tensorflow.python.training import training
import random
import re
from tensorflow.python.platform import gfile
import math
from six import iteritems
```

```
def triplet_loss(anchor, positive, negative, alpha):
    """Calculate the triplet loss according to the FaceNet paper

    Args:
      anchor: the embeddings for the anchor images.
      positive: the embeddings for the positive images.
      negative: the embeddings for the negative images.

    Returns:
      the triplet loss according to the FaceNet paper as a float tensor.
    """
    with tf.variable_scope('triplet_loss'):
        pos_dist = tf.reduce_sum(tf.square(tf.subtract(anchor, positive)), 1)
        neg_dist = tf.reduce_sum(tf.square(tf.subtract(anchor, negative)), 1)

        basic_loss = tf.add(tf.subtract(pos_dist,neg_dist), alpha)
        loss = tf.reduce_mean(tf.maximum(basic_loss, 0.0), 0)

    return loss

def center_loss(features, label, alfa, nrof_classes):
    """Center loss based on the paper "A Discriminative Feature Learning Approach for Deep Face Recognition"
       (http://ydwen.github.io/papers/WenECCV16.pdf)
    """
    nrof_features = features.get_shape()[1]
    centers = tf.get_variable('centers', [nrof_classes, nrof_features], dtype=tf.float32,
        initializer=tf.constant_initializer(0), trainable=False)
    label = tf.reshape(label, [-1])
    centers_batch = tf.gather(centers, label)
    diff = (1 - alfa) * (centers_batch - features)
    centers = tf.scatter_sub(centers, label, diff)
    with tf.control_dependencies([centers]):
        loss = tf.reduce_mean(tf.square(features - centers_batch))
    return loss, centers

def get_image_paths_and_labels(dataset):
    image_paths_flat = []
    labels_flat = []
    for i in range(len(dataset)):
        image_paths_flat += dataset[i].image_paths
        labels_flat += [i] * len(dataset[i].image_paths)
    return image_paths_flat, labels_flat

def shuffle_examples(image_paths, labels):
    shuffle_list = list(zip(image_paths, labels))
    random.shuffle(shuffle_list)
    image_paths_shuff, labels_shuff = zip(*shuffle_list)
    return image_paths_shuff, labels_shuff

def random_rotate_image(image):
    angle = np.random.uniform(low=-10.0, high=10.0)
    return misc.imrotate(image, angle, 'bicubic')

# 1: Random rotate 2: Random crop  4: Random flip  8:  Fixed image standardization  16: Flip
RANDOM_ROTATE = 1
RANDOM_CROP = 2
RANDOM_FLIP = 4
FIXED_STANDARDIZATION  = 8
FLIP = 16
def create_input_pipeline(input_queue, image_size, nrof_preprocess_threads, batch_size_placeholder):
    images_and_labels_list = []
    for _ in range(nrof_preprocess_threads):
        filenames, label, control = input_queue.dequeue()
    images = []
    for filename in tf.unstack(filenames):
        file_contents = tf.read_file(filename)
        image = tf.image.decode_image(file_contents, 3)
        image = tf.cond(get_control_flag(control[0], RANDOM_ROTATE),
                        lambda:tf.py_func(random_rotate_image, [image], tf.uint8),
                        lambda:tf.identity(image))
        image = tf.cond(get_control_flag(control[0], RANDOM_CROP),
                        lambda:tf.random_crop(image, image_size + (3,)),
                        lambda:tf.image.resize_image_with_crop_or_pad(image, image_size[0], image_size[1]))
        image = tf.cond(get_control_flag(control[0], RANDOM_FLIP),
                        lambda:tf.image.random_flip_left_right(image),
                        lambda:tf.identity(image))
        image = tf.cond(get_control_flag(control[0], FIXED_STANDARDIZATION),
                        lambda:(tf.cast(image, tf.float32) - 127.5)/128.0,
                        lambda:tf.image.per_image_standardization(image))
        image = tf.cond(get_control_flag(control[0], FLIP),
                        lambda:tf.image.flip_left_right(image),
                        lambda:tf.identity(image))
        #pylint: disable=no-member
        image.set_shape(image_size + (3,))
        images.append(image)
    images_and_labels_list.append([images, label])

image_batch, label_batch = tf.train.batch_join(
    images_and_labels_list, batch_size=batch_size_placeholder,
    shapes=[image_size + (3,), ()], enqueue_many=True,
    capacity=4 * nrof_preprocess_threads * 100,
    allow_smaller_final_batch=True)

return image_batch, label_batch
```

```
def get_control_flag(control, field):
    return tf.equal(tf.mod(tf.floor_div(control, field), 2), 1)


def _add_loss_summaries(total_loss):
    """Add summaries for losses.


    Generates moving average for all losses and associated summaries for
    visualizing the performance of the network.


    Args:
      total_loss: Total loss from loss().
    Returns:
      loss_averages_op: op for generating moving averages of losses.
    """
    # Compute the moving average of all individual losses and the total loss.
    loss_averages = tf.train.ExponentialMovingAverage(0.9, name='avg')
    losses = tf.get_collection('losses')
    loss_averages_op = loss_averages.apply(losses + [total_loss])


    # Attach a scalar summmary to all individual losses and the total loss; do the
    # same for the averaged version of the losses.
    for l in losses + [total_loss]:
        # Name each loss as '(raw)' and name the moving average version of the loss
        # as the original loss name.
        tf.summary.scalar(l.op.name +' (raw)', l)
        tf.summary.scalar(l.op.name, loss_averages.average(l))


    return loss_averages_op


def train(total_loss, global_step, optimizer, learning_rate, moving_average_decay, update_gradient_vars,
log_histograms=True):
    # Generate moving averages of all losses and associated summaries.
```

```
loss_averages_op = _add_loss_summaries(total_loss)

# Compute gradients.
with tf.control_dependencies([loss_averages_op]):
    if optimizer=='ADAGRAD':
        opt = tf.train.AdagradOptimizer(learning_rate)
    elif optimizer=='ADADELTA':
        opt = tf.train.AdadeltaOptimizer(learning_rate, rho=0.9, epsilon=1e-6)
    elif optimizer=='ADAM':
        opt = tf.train.AdamOptimizer(learning_rate, beta1=0.9, beta2=0.999, epsilon=0.1)
    elif optimizer=='RMSPROP':
        opt = tf.train.RMSPropOptimizer(learning_rate, decay=0.9, momentum=0.9, epsilon=1.0)
    elif optimizer=='MOM':
        opt = tf.train.MomentumOptimizer(learning_rate, 0.9, use_nesterov=True)
    else:
        raise ValueError('Invalid optimization algorithm')

    grads = opt.compute_gradients(total_loss, update_gradient_vars)

# Apply gradients.
apply_gradient_op = opt.apply_gradients(grads, global_step=global_step)

# Add histograms for trainable variables.
if log_histograms:
    for var in tf.trainable_variables():
        tf.summary.histogram(var.op.name, var)

# Add histograms for gradients.
if log_histograms:
    for grad, var in grads:
        if grad is not None:
            tf.summary.histogram(var.op.name + '/gradients', grad)
    # Track the moving averages of all trainable variables.
    variable_averages = tf.train.ExponentialMovingAverage(
        moving_average_decay, global_step)
    variables_averages_op = variable_averages.apply(tf.trainable_variables())

    with tf.control_dependencies([apply_gradient_op, variables_averages_op]):
        train_op = tf.no_op(name='train')

    return train_op

def prewhiten(x):
    mean = np.mean(x)
    std = np.std(x)
    std_adj = np.maximum(std, 1.0/np.sqrt(x.size))
    y = np.multiply(np.subtract(x, mean), 1/std_adj)
    return y

def crop(image, random_crop, image_size):
    if image.shape[1]>image_size:
        sz1 = int(image.shape[1]//2)
        sz2 = int(image_size//2)
        if random_crop:
            diff = sz1-sz2
            (h, v) = (np.random.randint(-diff, diff+1), np.random.randint(-diff, diff+1))
        else:
            (h, v) = (0,0)
        image = image[(sz1-sz2+v):(sz1+sz2+v),(sz1-sz2+h):(sz1+sz2+h),:]
    return image
def flip(image, random_flip):
    if random_flip and np.random.choice([True, False]):
        image = np.fliplr(image)
    return image

def to_rgb(img):
    w, h = img.shape
    ret = np.empty((w, h, 3), dtype=np.uint8)
    ret[:, :, 0] = ret[:, :, 1] = ret[:, :, 2] = img
    return ret

def load_data(image_paths, do_random_crop, do_random_flip, image_size, do_prewhiten=True):
    nrof_samples = len(image_paths)
    images = np.zeros((nrof_samples, image_size, image_size, 3))
    for i in range(nrof_samples):
        img = misc.imread(image_paths[i])
        if img.ndim == 2:
            img = to_rgb(img)
        if do_prewhiten:
            img = prewhiten(img)
        img = crop(img, do_random_crop, image_size)
        img = flip(img, do_random_flip)
        images[i,:,:,:] = img
    return images

def get_label_batch(label_data, batch_size, batch_index):
    nrof_examples = np.size(label_data, 0)
    j = batch_index*batch_size % nrof_examples
    if j+batch_size<=nrof_examples:
        batch = label_data[j:j+batch_size]
    else:
        x1 = label_data[j:nrof_examples]
        x2 = label_data[0:nrof_examples-j]
        batch = np.vstack([x1,x2])
    batch_int = batch.astype(np.int64)
    return batch_int

def get_batch(image_data, batch_size, batch_index):
    nrof_examples = np.size(image_data, 0)
    j = batch_index*batch_size % nrof_examples
    if j+batch_size<=nrof_examples:
        batch = image_data[j:j+batch_size,:,:,:]
    else:
        x1 = image_data[j:nrof_examples,:,:,:]
        x2 = image_data[0:nrof_examples-j,:,:,:]
        batch = np.vstack([x1,x2])
    batch_float = batch.astype(np.float32)
    return batch_float

def get_triplet_batch(triplets, batch_index, batch_size):
    ax, px, nx = triplets
    a = get_batch(ax, int(batch_size/3), batch_index)
    p = get_batch(px, int(batch_size/3), batch_index)
    n = get_batch(nx, int(batch_size/3), batch_index)
    batch = np.vstack([a, p, n])
    return batch

def get_learning_rate_from_file(filename, epoch):
    with open(filename, 'r') as f:
        for line in f.readlines():
            line = line.split('#', 1)[0]
            if line:
                par = line.strip().split(':')
                e = int(par[0])
                if par[1]=='-':
                    lr = -1
                else:
```

```
                lr = float(par[1])
            if e <= epoch:
                learning_rate = lr
            else:
                return learning_rate

class ImageClass():
    "Stores the paths to images for a given class"
    def __init__(self, name, image_paths):
        self.name = name
        self.image_paths = image_paths


    def __str__(self):
        return self.name + ', ' + str(len(self.image_paths)) + ' images'


    def __len__(self):
        return len(self.image_paths)

def get_dataset(path, has_class_directories=True):
    dataset = []
    path_exp = os.path.expanduser(path)
    classes = [path for path in os.listdir(path_exp) \
                    if os.path.isdir(os.path.join(path_exp, path))]
    classes.sort()
    nrof_classes = len(classes)
    for i in range(nrof_classes):
        class_name = classes[i]
        facedir = os.path.join(path_exp, class_name)
        image_paths = get_image_paths(facedir)
        dataset.append(ImageClass(class_name, image_paths))

    return dataset
```

```
def get_image_paths(facedir):
    image_paths = []
    if os.path.isdir(facedir):
        images = os.listdir(facedir)
        image_paths = [os.path.join(facedir,img) for img in images]
    return image_paths
def split_dataset(dataset, split_ratio, min_nrof_images_per_class, mode):
    if mode=='SPLIT_CLASSES':
        nrof_classes = len(dataset)
        class_indices = np.arange(nrof_classes)
        np.random.shuffle(class_indices)
        split = int(round(nrof_classes*(1-split_ratio)))
        train_set = [dataset[i] for i in class_indices[0:split]]
        test_set = [dataset[i] for i in class_indices[split:-1]]
    elif mode=='SPLIT_IMAGES':
        train_set = []
        test_set = []
        for cls in dataset:
            paths = cls.image_paths
            np.random.shuffle(paths)
            nrof_images_in_class = len(paths)
            split = int(math.floor(nrof_images_in_class*(1-split_ratio)))
            if split==nrof_images_in_class:
                split = nrof_images_in_class-1
            if split>=min_nrof_images_per_class and nrof_images_in_class-split>=1:
                train_set.append(ImageClass(cls.name, paths[:split]))
                test_set.append(ImageClass(cls.name, paths[split:]))
    else:
        raise ValueError('Invalid train/test split mode "%s"' % mode)
    return train_set, test_set
```

models.py:

```
from datetime import datetime
from attendance import app, db, login_manager
from flask_login import UserMixin

@login_manager.user_loader
def load_user(user_id):
        return User.query.get(int(user_id))

class User(db.Model,UserMixin):
        id = db.Column(db.Integer,primary_key=True)
        username = db.Column(db.String(20),unique=True,nullable=False)
        email = db.Column(db.String(20),unique=True,nullable=False)
        password = db.Column(db.String(20),nullable=False)
        def __repr__(self):
            return f"User('{self.username}','{self.email}')"

class Add(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        classname = db.Column(db.String(20))
        #students = db.Column(db.Integer, unique=True, nullable=False)
        coordinator = db.Column(db.String(30))
        co_email = db.Column(db.String(30))
        stuname = db.Column(db.String(30))
        regno = db.Column(db.Integer,unique=True)
        mobileno = db.Column(db.Integer,unique=True)
        # def get_my_form(self):
        #     from attendance.forms import AddForm
        #     return AddForm()
        # def insertion():
        #     form = self.get_my_form()
        #     num = form.students.data
```

```
            #   stu[num]
            #   regno[num]
            #   for i in range(num):
            def __repr__(self):
                return
f"Add('{self.classname}','{self.coordinator}','{self.co_email}','{self.stuname}','{self.regno}','{self.mobileno}')"
```

forms.py:

```
from flask_wtf import FlaskForm
from flask_wtf.file import FileField,FileAllowed
from flask_login import current_user
from wtforms import StringField, PasswordField, SubmitField, BooleanField, IntegerField
from wtforms.validators import DataRequired, Length, Email, EqualTo, ValidationError
from attendance.models import User,Add


class RegistrationForm(FlaskForm):
            username = StringField('Username',validators=[DataRequired(),Length(min=2,max=20)])
            email = StringField('Email', validators=[DataRequired(),Email()])
            password = PasswordField('Password', validators=[DataRequired()])
            confirm_password = PasswordField('Confirm Password',
validators=[DataRequired(),EqualTo('password')])
            submit = SubmitField('Register')
            def validate_username(self,username):
                user = User.query.filter_by(username=username.data).first()
                if user:
                        raise ValidationError('That username is taken. Please choose a different one')


            def   validate_email(self,email):
                user = User.query.filter_by(email=email.data).first()
                if user:
                        raise ValidationError('That email is taken. Please choose a different one')
```

```
class LoginForm(FlaskForm):
                    email = StringField('Email',validators=[DataRequired(),Email()])
                    password = PasswordField('Password',validators=[DataRequired()])
                    remember = BooleanField('Remember Me')
                    submit = SubmitField('Login')
```

index.html:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<meta charset="utf-8">
<title>AMFR</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
<link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet"
integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN"
crossorigin="anonymous">
<link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
<script type="text/javascript">
$(window).on('load',function(){
$('.preloader').addClass('complete')
})
</script>
</head>
<body data-spy="scroll" date-target=".navbar" data-offset="50">
<header>
<nav class="navbar navbar-expand-lg navbar-light bg-light fixed-top">
<div class="container">
<a class="navbar-brand" href="#">AMFR</a>
<button class="navbar-toggler collapsed" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
```

```
<span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav ml-lg-auto">
<li class="nav-item">
<a class="nav-link" href="#home">Home <span class="sr-only">(current)</span></a>
</li>
<li class="nav-item">
<a class="nav-link" href="#explore">Explore</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#launch">Launch</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#blog">Blog</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#contact">Contact</a>
</li>
{% if current_user.is_authenticated %}
<li class="nav-item">
<a class="nav-item nav-link" href="{{ url_for('logout') }}">Logout</a>
</li>
{% else %}
<li>
<a class="nav-item nav-link" href="{{ url_for('login') }}">Login</a>
</li>
<li>
<a class="nav-item nav-link" href="{{ url_for('register') }}">Register</a>
</li>
{% endif %}
```

```
</ul>
</div>
</div>
</nav>
</header>
<div class="jumbotron jumbotron-fluid height100p banner" id="home">
<div class="container h100">
<div class="contentBox h100">
<div>
<h1 data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">Attendance Monitoring</h1>
<p data-aos="fade-up" data-aos-duration="1000" data-aos-delay="500">The application uses google facenet to
recognize students and mark attendance to the students. A high defintion picture has to be uploaded, so that faces are
sharp to recoginze.</p>
</div>
</div>
</div>
</div>
<section class="sec1" id="explore">
<div class="container">
<div class="row">
<div class="offset-sm-2 col-sm-8">
<div class="headerText text-center">
<h2 data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">Explore AMRF</h2>
<p data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.
</p>
</div>
</div>
</div>
```

```
<div class="row">
<div class="col-sm-4">
<div class="placeBox" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">
<div class="imgBx">
<img src="{{ url_for('static', filename='images/img1.jpg') }}" class="img-fluid">
</div>
<div class="content text-center">
<a href="#" class="btn btnD1">About Project</a>
</div>
</div>
</div>
<div class="col-sm-4">
<div class="placeBox" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="250">
<div class="imgBx">
<img src="{{ url_for('static', filename='images/img2.jpg') }}" class="img-fluid">
</div>
<div class="content text-center">
<a href="{{ url_for('add_class') }}" class="btn btnD1">Add Class</a>
</div>
</div>
</div>
<div class="col-sm-4">
<div class="placeBox" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="500">
<div class="imgBx">
<img src="{{ url_for('static', filename='images/img3.png') }}" class="img-fluid">
</div>
<div class="content text-center">
<a href="{{ url_for('take_attendance')}}" class="btn btnD1">Take Attendance</a>
</div>
</div>
</div>
</div>
```

```
</div>
</section>
<section class="sec2" id="launch">
<div class="container h100">
<div class="contentBox h100">
<div>
<h1 data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">AI is Everywhere</h1>
<p data-aos="fade-up" data-aos-duration="1000" data-aos-delay="100">
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.
</p>
<a href="#" class="btn btnD3" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="100">Read
More</a>
</div>
</div>
</div>
</section>
<section class="blog" id="blog">
<div class="container">
<div class="row">
<div class="offset-sm-2 col-sm-8">
<div class="headerText text-center">
<h2 data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">About Our Latest Post</h2>
<p data-aos="fade-up" data-aos-duration="1000" data-aos-delay="100">With the exploding amount of mobile
traffic data and unprecedented demands of computing, executing the ever increasingly complex applications in
resource-constrained mobile devices becomes more and more challenging.</p>
<a href="{{ url_for('dmc') }}" class="btn btnD3" data-aos="fade-up" data-aos-duration="1000" data-aos-
delay="100">Read More</a>
</div>
</div>
</div>
```

```
<div class="row">

<div class="col-sm-6">

<div class="blogpost" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="100">

<div class="imgBx">

<img src="{{ url_for('static', filename='images/comv.png') }}" class="img-fluid">

</div>

<div class="content">

<h1>Computer vision: A Vision Of Future</h1>

<p>Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level
understanding from digital images or videos......</p>

<a href="{{ url_for('cv') }}" class="btn btnD2">Read More</a>

<div class="clearfix"></div>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="blogpost" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="500">

<div class="imgBx">

<img src="{{ url_for('static', filename='images/mob.jpg') }}" class="img-fluid">

</div>

<div class="content">

<h1>Deep Mobile Computing</h1>

<p>With the exploding amount of mobile traffic data and unprecedented demands of computing, executing the ever
increasingly complex applications in resource......</p>

<a href="{{ url_for('dmc') }}" class="btn btnD2">Read More</a>

<div class="clearfix"></div>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="blogpost" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="100">

<div class="imgBx">
```

```
<img src="{{ url_for('static', filename='images/flask.png') }}" class="img-fluid">

</div>

<div class="content">

<h1>Flask: A Developer’s Tool</h1>

<p>Flask is a micro web framework written in Python. It is classified as a microframework because it does not
require particular tools or libraries......</p>

<a href="{{ url_for('flsk') }}" class="btn btnD2">Read More</a>

<div class="clearfix"></div>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="blogpost" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="500">

<div class="imgBx">

<img src="{{ url_for('static', filename='images/sqlytdb.png') }}" class="img-fluid">

</div>

<div class="content">

<h1>SQLite: A database</h1>

<p>SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to man y
other database management systems......</p>

<a href="{{ url_for('sqlyt') }}" class="btn btnD2">Read More</a>

<div class="clearfix"></div>

</div>

</div>

</div>

</div>

</div>

</section>

<section class="contact" id="contact">

<div class="container">

<div class="row">

<div class="col-sm-12">
```

```
<div class="headerText text-center">

<h2 data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">Contact Us</h2>

<p data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">for more information regarding the
application you can contact us</p>

</div>

</div>

</div>

<div class="row clearfix">

<div class="offset-sm-2 col-sm-8">

<form data-aos="fade-up" data-aos-duration="1000" data-aos-delay="200">

<div class="form-group">

<label>Name</label>

<input type="text" name="" class="form-control">

</div>

<div class="form-group">

<label>Email</label>

<input type="text" name="" class="form-control">

</div>

<div class="form-group">

<label>Phone</label>

<input type="text" name="" class="form-control">

</div>

<div class="form-group">

<label>Feedback</label>

<textarea class="form-control textarea" name=""></textarea>

</div>

<div class="form-group text-center">

<br>

<button class="btn btnD4">Send</button>

</div>

</form>

</div>
```

```
</div>
</div>
</section>
<footer>
<div class="container">
<div class="row">
<div class="col-sm-12">
<ul class="sci">
<li>
<a href="#">
<i class="fa fa-facebook"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-twitter"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-google-plus"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-linkedin"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-instagram"></i><!--use fontawesome icon -->
</a>
```

```
</li>

</ul>

<p class="cpryt">© Copyright 2020 AMFR | Template by <a href="#">sairaj_priyatam</a></p>

</div>

</div>

</div>

</footer>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>

<script src="https://unpkg.com/aos@next/dist/aos.js"></script>

<script>

AOS.init();

</script>

<script type="text/javascript">

$(document).scroll(function(){

$('.navbar').toggleClass('scrolled', $(this).scrollTop() > $('.navbar').height());

});

</script>

</body>

</html>
```

login.html:

```
<!DOCTYPE html>

<html>

<head>

<title>Login Page</title>

<meta name="viewport" content="width=device-width,initial-scale=1.0">
```

```
<meta charset="utf-8">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

<link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet"
integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN"
crossorigin="anonymous">

<link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />

<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='login.css') }}">

</head>

<body>

{% block content %}

<header>

<nav class="navbar navbar-expand-lg navbar-light bg-light fixed-top">

<div class="container">

<a class="navbar-brand" href="#">AMFR</a>

<button class="navbar-toggler collapsed" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav ml-lg-auto">

<li class="nav-item">

<a class="nav-link" href="{{ url_for('index') }}">Home <span class="sr-only">(current)</span></a>

</li>

{% if current_user.is_authenticated %}

<li class="nav-item">

<a class="nav-item nav-link" href="{{ url_for('logout') }}">Logout</a>

</li>

{% else %}

<li>

<a class="nav-item nav-link" href="{{ url_for('register') }}">Register</a>

</li>
```

```
{% endif %}
</ul>
</div>
</div>
</nav>
</header>
<section class="logincontact" id="login">
<div class="container">
<div class="row">
<div class="col-sm-12">
<div class="headerText text-center">
<h1 data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">LOGIN</h1>
</div>
</div>
</div>
<div class="row clearfix">
<div class="offset-sm-2 col-sm-8">
<form method="POST" action="" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="200">
{{ form.hidden_tag() }}
<fieldset class="form-group">
<div class="form-group">
<label>Email</label>
{% if form.email.errors %}
{{ form.email(class="form-control") }}
<div class="invalid-feedback">
{% for error in form.email.errors %}
<span>{{ error }}</span>
{% endfor %}
</div>
{% else %}
{{ form.email(class="form-control") }}
{% endif %}
```

```
</div>
<div class="form-group">
<label>Password</label>
{% if form.password.errors %}
{{ form.password(class="form-control") }}
<div class="invalid-feedback">
{% for error in form.password.errors %}
<span>{{ error}}</span>
{% endfor %}
</div>
{% else %}
{{ form.password(class="form-control") }}
{% endif %}
<div class="form-check">
<br>
{{ form.remember(class="form-check-input") }}
<small class="text-muted">Remember Me</small>
</div>
</div>
</fieldset>
<div class="form-group text-center">
{{ form.submit(class="btn btnD4") }}
</div>
<div class="form-group text-center">
<small class="text-muted ml-2">
<a href="#">Forgot Password?</a>
</small>
</div>
<div class="form-group text-center">
<small class="text-muted">
Need An Account? <a class="ml-2" href="{{ url_for('register') }}">Register</a>
</small>
```

```
</div>
</form>
</div>
</div>
</div>
</section>
{% endblock %}
<footer>
<div class="container">
<div class="row">
<div class="col-sm-12">
<ul class="sci">
<li>
<a href="#">
<i class="fa fa-facebook"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-twitter"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-google-plus"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-linkedin"></i><!--use fontawesome icon -->
</a>
</li>
```

```
<li>

<a href="#">

<i class="fa fa-instagram"></i><!--use fontawesome icon -->

</a>

</li>

</ul>

<p class="cpryt">© Copyright 2020 AMFR | Template by <a href="#">sairaj_priyatam</a></p>

</div>

</div>

</div>

</footer>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>

<script src="https://unpkg.com/aos@next/dist/aos.js"></script>

<script>

AOS.init();

</script>

<script type="text/javascript">

$(document).scroll(function(){

$('.navbar').toggleClass('scrolled', $(this).scrollTop() > $('.navbar').height());

});

</script>

</body>

</html>
```

add_class.html:

```
<!DOCTYPE html>
```

```
<html>

<head>

<title>Add-New-Class</title>

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<meta charset="utf-8">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

<link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet"
integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN"
crossorigin="anonymous">

<link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />

<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='add_class.css') }}">

</head>

<body>

{% block content %}

<header>

<nav class="navbar navbar-expand-lg navbar-light bg-light fixed-top">

<div class="container">

<a class="navbar-brand" href="#">AMFR</a>

<button class="navbar-toggler collapsed" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav ml-lg-auto">

<li class="nav-item">

<a class="nav-link" href="{{ url_for('index') }}">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="{{ url_for('take_attendance') }}">Take Attendance<span class="sr-
only">(current)</span></a>

</li>
```

```
{% if current_user.is_authenticated %}
<li class="nav-item">
<a class="nav-item nav-link" href="{{ url_for('logout') }}">Logout</a>
</li>
{% else %}
<li>
<a class="nav-item nav-link" href="{{ url_for('login') }}">Login</a>
</li>
<li>
<a class="nav-item nav-link" href="{{url_for('register') }}">Register</a>
</li>
{% endif %}
</ul>
</div>
</div>
</nav>
</header>
<section class="addclasscontact">
<div class="container">
<div class="row">
<div class="col-md-12">
<div class="headerText text-center">
<h1 data-aos="fade-up" data-aos-duration="1000" data-aos-delay="0">CREATE NEW CLASS</h1>
</div>
</div>
</div>
<div class="row clearfix">
<div class="offset-sm-2 col-sm-8">
<form action="/add" enctype="multipart/form-data" method="POST" data-aos="fade-up" data-aos-duration="1000"
data-aos-delay="200">
<div class="form-row">
<div class="form-group col-md-6">
```

```
<label>Class Name</label>
</div>
<div class="form-group col-md-4">
<label>No of Students</label>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<input class="form-control" type="text" name="classname">
</div>
<div class="form-group col-md-4">
<input class="form-control" type="text" name="noofstu" id="noofstu">
</div>
<div class="form-group col-md-2">
<input class="btn btnD4 text-center" type="button" value="Generate Feilds" id="addFields"/>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<label>Co-ordinator Name</label>
</div>
<div class="form-group col-md-6">
<label>Co-ordinator Email</label>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<input class="form-control" type="text" name="coorname">
</div>
<div class="form-group col-md-6">
<input class="form-control" type="text" name="cooremail">
</div>
```

```
</div>
<div class="form-row" id="pre">

</div>
<div class="form-row">
<div class="form-group col-md-6">
<button class="btn btnD4" type="submit" name="create">Create</button>
</div>
</div>
</form>
</div>
</div>
</div>
</section>
{% endblock %}
<footer>
<div class="container">
<div class="row">
<div class="col-sm-12">
<ul class="sci">
<li>
<a href="#">
<i class="fa fa-facebook"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
<i class="fa fa-twitter"></i><!--use fontawesome icon -->
</a>
</li>
<li>
<a href="#">
```

```
<i class="fa fa-google-plus"></i><!--use fontawesome icon -->

</a>

</li>

<li>

<a href="#">

<i class="fa fa-linkedin"></i><!--use fontawesome icon -->

</a>

</li>

<li>

<a href="#">

<i class="fa fa-instagram"></i><!--use fontawesome icon -->

</a>

</li>

</ul>

<p class="cpryt">© Copyright 2020 AMFR | Template by <a href="#">sairaj_priyatam</a></p>

</div>

</div>

</div>

</footer>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>

<script src="https://unpkg.com/aos@next/dist/aos.js"></script>

<script>

AOS.init();

</script>

<script type="text/javascript">

$(document).scroll(function(){
```

```
$('.navbar').toggleClass('scrolled', $(this).scrollTop() > $('.navbar').height());

});

$(document).ready(function(){

$("#addFields").on("click",function(){

$number_of_columns = $("#noofstu").val();

$number_of_columns = parseInt($number_of_columns);

for($i=0; $i<$number_of_columns; $i++)

{

$new_element ='<div class="form-group col-md-4"><label>Student Name</label><input class="form-control"
type="text" name="namefields[]"></div><div class="form-group col-md-4"><label>Phone Number</label><input
class="form-control" type="text" name="phonefields[]"></div><div class="form-group col-md-4"><label>Roll
Number</label><input class="form-control" type="text" name="rollfields[]"></div>';

$("#pre").append($new_element);

}

});

});

</script>

</body>

</html>
```

style.css:

```
@import url("https://fonts.googleapis.com/css?family=Open+Sans")

*

{

                outline: none !important;

}

html

{

                scroll-behavior: smooth;

}

body

{

                font-variant: 'Poppins', sans-serif;
```

```
}
header
{
                              position: relative;
                              z-index: 1000000;
}
p
{
                              font-size: 1.15rem;
}
.navbar
{
                              background: transparent !important;
                              transition: 0.5s;
}
.navbar.scrolled
{
                              background: #000 !important;
                              transition: 0.5s;
}
.navbar-light .navbar-nav .active>.nav-link, .navbar-light .navbar-nav .nav-link.active, .navbar-light .navbar-nav
.nav-link.show, .navbar-light .navbar-nav .show>.nav-link
{
    color: #fff;
    border-bottom: 2px solid #fff;
    border-radius: none;
    border-width: 2px;
    padding: 5px;
    width: auto;
    text-align: center;
    margin-right: 6px;
```

```
}
.navbar-light .navbar-nav .active>.nav-link:hover
{
                    color: #000;
}
.navbar-light .navbar-nav .nav-link
{
  color: #fff;
  border-bottom: 2px solid #000;
  border-radius: none;
  border-width: 2px;
  padding: 5px;
  width: auto;
  text-align: center;
  margin-right: 6px;
}
.navbar-light .navbar-nav .nav-link:hover,
.navbar-light .navbar-nav .nav-link:focus
{
                    color: #fff;
                    background: #000;
  border-bottom: 2px solid #fff;
  border-radius: none;
  border-width: 2px;
  padding: 5px;
  width: auto;
  text-align: center;
  margin-right: 6px;
  transition: 0.5s;
}
.navbar-light .navbar-brand,
.navbar-light .navbar-brand:hover,
.navbar-light .navbar-brand:focus,
.navbar-light .navbar-brand:visited
{
    color: rgba(255,255,255,1);
    font-size: 1.7rem;
    font-weight: 600;
}
.banner
{
                              position: relative;
                              top: 0;
                              width: 100%;
                              height: 100vh;
                              background-size: cover;
                              background: url(images/banner2.jpg);
                              margin-bottom: 0;
}
.banner:before,
.sec2:before,
.contact:before
{
                              content: '';
                              position: absolute;
                              top: 0;
                              left: 0;
                              width: 100%;
                              height: 400px;
                              background: linear-gradient(#000,transparent);
                              pointer-events: none;


}
.banner:after,
.sec2:after,
.contact:after
{
                content: '';
                position: absolute;
                bottom: 0;
                left: 0;
                width: 100%;
                height: 400px;
                background: linear-gradient(360deg,#000,transparent);
                pointer-events: none;

}
.height100p
{
                height: 100vh;
}
.h100
{
                height: 100%;
}
.contentBox
{
                position: relative;
                display: flex;
                justify-content: center;
                align-items: center;
                text-align: center;
                width: 100%;
                height: 100%;
                z-index: 10;
}
```

```
.contentBox h1
{
                font-weight: 800;
                color: #fff;
                text-transform: uppercase;
                font-size: 5rem;
                padding-bottom: 15px;
}
.contentBox p
{
                color: #fff;
                font-size: 1.3rem;
}
.sec1
{
                padding: 100px 0;
                background: #000;
}
.headerText
{
                color: #fff;
}
.headerText h2
{
                font-size: 2.5rem;
                margin-bottom: 20px;
}
.placeBox
{
                position: relative;
                max-width: 300px;
                height: 400px;
```

```
                        margin: 0 auto;
                        background: #000;
                        margin-top: 30px;
                        border: 2px solid #fff;
                        border-radius: 7px;
}
.placeBox .imgx
{
                        width: 100%;
                        height: 100%;

}
.placeBox .imgx img
{
                        object-fit: cover;
                        width: 100%;
                        height: 100%;

}
.btnD1
{
                        position: relative;
                        color: #fff;
   border-top: 2px solid #fff;
   border-bottom: 2px solid #fff;
   border-width: 2px;
   padding: 5px;
   width: 130px;
   margin-top: 30px;
   text-align: center;
}
.btnD1:hover,
.btnD1:focus
{
                    position: relative;
                    color: #fff;
                    background: #000;
                    border-top: 2px solid #fff;
  border-bottom: 2px solid #fff;
  border-width: 2px;
  padding: 5px;
  width: 140px;
  margin-top: 30px;
  text-align: center;
}
.sec2
{
                    position: relative;
                    background: url(images/banner2.jpg);
                    background-size: cover;
                    height: 100vh;
}
.sec2 h2
{
                    font-weight: 800;
                    color: #fff;
                    text-transform: uppercase;
                    font-size: 4.5rem;
}
.btnD3
{
                    border: 2px solid #fff;
                    color: #fff;
                    font-size: 1.15rem;
```

```
                outline: none;
                font-weight: 500;
                margin-top: 20px;
}
.btnD3:hover,
.btnD3:focus
{
                border: 2px solid #fff;
                color: #000;
                background: #fff;
                font-size: 1.15rem;
                outline: none;
                font-weight: 500;
                margin-top: 20px;
}
.blog
{
                position: relative;
                padding: 100px 0;
                background: #000;
}
.blogpost
{
                position: relative;
                width: 100%;
                margin: 0 auto;
                background: #000;
                margin-top: 30px;
                border: 5px solid rgba(255,255,255,.2);
                border-radius: 7px;
}
.blogpost .imgBx
```

```
{
        height: 400px;
}
.blogpost .imgBx img
{
        width: 100%;
        height: 100%;
        object-fit: cover;
}
.blogpost .content
{
        padding: 15px;
}
.blogpost .content h1
{
        margin: 5;
        color: #fff;
        padding: 0 0 20px;
        font-weight: 600;
        font-size: 1.5rem;
}
.blogpost .content p
{
        margin: 5;
        color: #fff;
        padding: 0 0 20px;
        font-weight: 600;
        font-size: 1rem;
}
.btnD2
{
        position: relative;
```

```
                                color: #fff;
    border: 2px solid #fff;
    border-radius: 5px;
    border-width: 2px;
    padding: 5px;
    width: 120px;
    text-align: center;
    margin-right: 6px;
    float: right;
}
.btnD2:hover,
.btnD2:focus
{
                                position: relative;
                                color: #000;
                                background: #fff;
    border: 2px solid #fff;
    border-radius: 5px;
    border-width: 2px;
    padding: 5px;
    width: 120px;
    text-align: center;
    margin-right: 6px;
    float: right;
}
.contact
{
                                position: relative;
                                padding: 100px 0;
                                background: url(images/banner2.jpg);
                                background-size: cover;
                                min-height: 100vh;
```

```
}
.contact form
{
                position: relative;
                z-index: 1000;
}
.contact .form-control
{
                background: transparent;
                border: none;
                border-bottom: 2px solid rgba(255,255,255,.2);
                height: 50px;
                border-radius: 0;
                background: rgba(0,0,0,.6);
                color: #fff;
                font-size: 1.25rem;
}
.contact .form-control:hover,
.contact .form-control:focus
{
                border: none;
                border-bottom: 2px solid rgba(255,255,255,1);
                box-shadow: none;
}
.contact .textarea
{
                min-height: 100px;
}
.contact label
{
                color: #fff;
                font-size: 1.25rem;
```

```
}
.btnD4
{
                                  position: relative;
                                  color: #fff;
                                  background: #000;
    border: 2px solid rgba(255,255,255,.2);
    border-radius: 5px;
    border-width: 2px;
    padding: 5px;
    width: 120px;
    text-align: center;
    margin-right: 6px;
}
.btnD4:hover,
.btnD4:focus
{
                                  position: relative;
                                  color: #000;
                                  background: #fff;
    border: 2px solid rgba(255,255,255,1);
    border-radius: 5px;
    border-width: 2px;
    padding: 5px;
    width: 120px;
    text-align: center;
    margin-right: 6px;
}
footer
{
                                  background: #000;
                                  padding: 50px 0 0;
```

```
}
.sci
{
                    margin: 0;
                    padding: 0;
                    display: flex;
                    justify-content: center;
                    align-items: center;
}
.sci li
{
                    list-style: none;
                    margin: 0 20px;
}
.sci li a
{
                    color: #777;
                    font-size: 2em;
                    transition: 0.5s;
}
.sci li a:hover,
.sci li a:focus
{
                    color: #fff;
}
.cpryt
{
                    margin-top: 20px;
                    text-align: center;
                    color: #777;
}
@media (min-width: 992px)
```

```
{
                    .navbar-expand-lg .navbar-nav .nav-link
                    {
                      padding-right: .5rem;
                      padding-left: .5rem;
                    }
}
@media (max-width: 992px)
{
                    .navbar-light .navbar-toggler
                    {
                        background: #fff;
                        border-radius:2px;
                    }
                    .banner
                    {
                        min-height: 100vh;
                        padding: 20px 0;
                    }
                    .contentBox h1
                    {
                        font-size: 2rem;
                        padding: 20px 0;
                    }
                    .contentBox p
                    {
                        font-size: 1rem;
                        padding: 20px 0;
                    }
                    .navbar-expand-lg .navbar-nav .nav-link
                    {
                      padding-right: .5rem;
```

```
    padding-left: .5rem;
}
.sec1
{
    padding: 20px 0;
}
.headerText h2
{
    font-size: 2rem;
}
.blog
{
    padding: 20px 0;
}
.blogpost .imgBx
{
height: 200px;
}
.contact
{
    padding: 20px 0;
}
.contact .form-control
{
        height: 40px;
        border: 1px solid #fff;
}
footer
{
    background: #000;
    padding: 0 0 20px;
}
```

```
                .sci li
                {
                  list-style: none;
                  margin: 0 10px;
                }
                .sci li a
                {
                    color: #777;
                    font-size: 1.5rem;
                    transition: 0.5s;
                }
                .cpryt
                {
                  margin-top: 20px;
                  text-align: center;
                  color: #777;
                  font-size: 0.75rem;
                }
                .sec2
                {
                    height: auto;
                    padding: 40px 0;
                }
}
```

login.css:

```
@import url("https://fonts.googleapis.com/css?family=Open+Sans")
*
{
                outline: none !important;
}
html
```

```
{
                scroll-behavior: smooth;
}
body
{
                font-variant: 'Poppins', sans-serif;
}
.navbar
{
  background: transparent !important;
  transition: 0.5s;
}
.navbar.scrolled
{
  background: #000 !important;
  transition: 0.5s;
}
.navbar-light .navbar-nav .active>.nav-link, .navbar-light .navbar-nav .nav-link.active, .navbar-light .navbar-nav
.nav-link.show, .navbar-light .navbar-nav .show>.nav-link
{
  color: #fff;
  border-bottom: 2px solid #fff;
  border-radius: none;
  border-width: 2px;
  padding: 5px;
  width: auto;
  text-align: center;
  margin-right: 6px;

}
.navbar-light .navbar-nav .active>.nav-link:hover
{
  color: #000;
}
.navbar-light .navbar-nav .nav-link
{
  color: #fff;
  border-bottom: 2px solid #000;
  border-radius: none;
  border-width: 2px;
  padding: 5px;
  width: auto;
  text-align: center;
  margin-right: 6px;
}
.navbar-light .navbar-nav .nav-link:hover,
.navbar-light .navbar-nav .nav-link:focus
{
  color: #fff;
  background: #000;
  border-bottom: 2px solid #fff;
  border-radius: none;
  border-width: 2px;
  padding: 5px;
  width: auto;
  text-align: center;
  margin-right: 6px;
  transition: 0.5s;
}
.navbar-light .navbar-brand,
.navbar-light .navbar-brand:hover,
.navbar-light .navbar-brand:focus,
.navbar-light .navbar-brand:visited
{
  color: rgba(255,255,255,1);
  font-size: 1.7rem;
  font-weight: 600;
}
.logincontact
{
  position: relative;
  padding: 100px 0;
  justify-content: center;
  display: flex;
  align-items: center;
  background: url(images/banner2.jpg);
  background-size: cover;
  min-height: 100vh;
}
.logincontact:before
{
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 400px;
  background: linear-gradient(#000,transparent);
  pointer-events: none;

}
.logincontact:after
{
  content: '';
  position: absolute;
  bottom: 0;
  left: 0;
  width: 100%;
  height: 400px;
  background: linear-gradient(360deg,#000,transparent);
  pointer-events: none;

}
.logincontact form
{
  position: relative;
  z-index: 1000;
}
.logincontact .form-control
{
  background: transparent;
  border: none;
  border-bottom: 2px solid rgba(255,255,255,.2);
  height: 50px;
  min-width: 400px;
  align-content: center;
  align-items: center;
  border-radius: 0;
  background: rgba(0,0,0,.6);
  color: #fff;
  font-size: 1.25rem;
}
.logincontact .form-control:hover,
.logincontact .form-control:focus
{
  border: none;
  border-bottom: 2px solid rgba(255,255,255,1);
  box-shadow: none;
```

```
}
.logincontact h1
{
    color: #fff;
}
.logincontact label
{
    color: #fff;
    font-size: 1.25rem;
}
.btnD4
{
    position: relative;
    color: #fff;
    background: transparent;
    border: 2px solid rgba(255,255,255,.2);
    border-radius: 5px;
    border-width: 2px;
    padding: 5px;
    height: auto;
    width: 120px;
    text-align: center;
    margin-right: 6px;
}
.btnD4:hover,
.btnD4:focus
{
    position: relative;
    color: #000;
    background: #fff;
    border: 2px solid rgba(255,255,255,1);
    border-radius: 5px;
    border-width: 2px;
    padding: 5px;
    height: auto;
    width: 120px;
    text-align: center;
    margin-right: 6px;
}
footer
{
    background: #000;
    padding: 500px 0 0;
}
.sci
{
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
}
.sci li
{
    list-style: none;
    margin: 0 20px;
}
.sci li a
{
    color: #777;
    font-size: 2em;
    transition: 0.5s;
}
.sci li a:hover,
.sci li a:focus
{
   color: #fff;
}
.cpryt
{
   margin-top: 20px;
   text-align: center;
   color: #777;
}
@media (min-width: 992px)
{
   .navbar-expand-lg .navbar-nav .nav-link
   {
      padding-right: .5rem;
      padding-left: .5rem;
   }
}
@media (max-width: 992px)
{
   .navbar-light .navbar-toggler
   {
      background: #fff;
      border-radius:2px;
   }
   .navbar-expand-lg .navbar-nav .nav-link
   {
      padding-right: .5rem;
      padding-left: .5rem;
   }
   .headerText h2
   {
      font-size: 2rem;
   }
   .logincontact
   {
      padding: 20px 0;
   }
   .logincontact .form-control
   {
      height: 40px;
      border: 1px solid #fff;
   }
   footer
   {
      background: #000;
      padding: 0 0 20px;
   }
   .sci li
   {
      list-style: none;
      margin: 0 10px;
   }
   .sci li a
   {
      color: #777;
      font-size: 1.5rem;
      transition: 0.5s;
   }
   .cpryt
   {
      margin-top: 20px;
      text-align: center;
      color: #777;
```

```
        font-size: 0.75rem;
    }
}
```