FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 8 No. 6, December, (Special Issue) 2024, pp 7 - 16
DOI: https://doi.org/10.33003/fjs-2024-0806-2773

A TWO-STEP AUTHENTICATION FACIAL RECOGNITION SYSTEM FOR AUTOMATED ATTENDANCE TRACKING

***Osazuwa-Ojo, Victory Osaruese and Elaigwu, V. O.**

Department of Electrical/Electronics Engineering, Benson Idahosa University, Okha, Edo State, Nigeria.

*Corresponding authors' email: victoryosazuwaojo@gmail.com

ABSTRACT
This study addresses the need for efficient, automated attendance systems through the design of a facial recognition application. Manual attendance systems are slow, error-prone and the retrieval of old records can be tedious. Universally assessable technological developments such as facial recognition software can easily solve these problems. However, the vast amount of computational resources required for its implementation has posed a limitation to its wide adoption. This study presents a two-step approach to resolve these challenges. By leveraging a faster, less-powerful model, as the first step, the workload of facial recognition can be distributed to save time and computational cost. A more powerful machine learning model is applied as the second step, deployed for tasks that are too complex for the first model to handle. The two-step authentication process will also reduce the occurrences of false negatives. Face_recognition, a python library is used for detection and encoding of face images read using python's opencv library from an IP webcam. A flask application demonstrates this facial recognition functionality. The database connection and communication are accomplished using flask_sqlalchemy. A graphical user interface (web application) is used to interact with users on a high level, showing saved images of logged personnel and their times of entry. The system has a maximum accuracy of 98.78% and precision of 98.82% from tests. This shows its potential for application on a wider scale, with some added improvements such as cloud deployment and larger datasets.

Keywords: Facial recognition attendance system, Facial recognition, Flask application, Python, Opencv

INTRODUCTION

Facial recognition is an identity verification technique that uses the individual's face as an identifier. It is a form of biometric verification. Facial recognition technology generally operates by following the following steps:
Face detection: The camera detects images of faces and isolates them. These images could be alone or in a crowd. Different orientations of the face may also be detected.
Face analysis: These images detected in the previous step are captured and analyzed. Several key facial landmarks that are used to distinguish individual faces are identified in this stage.
Converting the image to data: A unique faceprint, which in essence is a set of digital information based on the facial features of a given individual is produced from the captured face.
Finding a match: At this stage, comparison occurs. The faceprint is compared with several other faceprints of known individuals in a database. When a match is made, the identity of the person is confirmed (Kaspersky, n.d.).
Face recognition is an application of face detection. Beyond just detecting the presence of a human face, it applies algorithms on the detected face to determine its identity by comparison with a database of stored records (Barney N., n.d).
In the system implemented using flask, face detection, analysis and encoding is done by python's Face_recognition library. For the comparison of encodings (faceprints), a Support Vector Classifier is used. The InsightFace python package using the default model, buffalo_l is used as a second stage, where it carries out its own face analysis, encoding (interpreting face data) and comparison (finding a match).
The history of facial recognition as a medium of identification and security purposes stems as far back as 1871, when physical evidence in the form of a comparison of two photographs was presented and heard in an English court in the hearing of the 'Tichborne Claimant' case (Glenn Porter and Greg Doran, 2000).
Since then, technological advancements in facial recognition have improved its precision and application. In 1964, in the early stages of its research, American researchers studied facial recognition computer programming, imagining a semi-automatic method, where operators entered twenty computer measures, such as the size of the mouth or eyes. Some 13 years later, in 1977, the system was improved by adding 21 additional markers (e.g. lip width, hair color). Later on, in 1988, Artificial Intelligence was introduced to develop previously used theoretical tools. Mathematical methods were used to interpret, manipulate, and simplify images independent of human markers. In 1991, Alex Pentland and Matthew Turk of the Massachusetts Institute of Technology (MIT) presented the first successful example of facial recognition technology, Eigenfaces, which uses the statistical Principal Component Analysis (PCA) method. In order to encourage the advancement of the technology, the Defense Advanced Research Project Agency (DARPA) developed the Face Recognition Technology (FERET) program, which provided a database of 2400 images for 850 persons in 1998. In the 2000s, The Face Recognition Grand Challenge (FRGC) competition was launched to encourage and develop face recognition technology designed to support existent facial recognition initiatives (2005) and in 2011, there was a marked acceleration of facial recognition technology due to deep learning, a machine learning method based on artificial neural networks.
In 2014, Facebook's internal algorithm DeepFace approaches performance of the human eye, with an accuracy near to 97%. Apple, in its new updates introduced a facial recognition application where its implementation has extended to retail and banking. Mastercard developed the Selfie Pay, a facial recognition framework for online transactions. In China, from 2019, people who want to buy a new phone consent to have their faces checked by the operator and the police use a smart monitoring system based on live facial recognition. This

system was able to locate a suspect from a crowd of 50000 persons at a concert (Adjabi, I.et al, 2020).

Several techniques exist for facial recognition.

Kortli, et al (2020) also surveys the techniques for facial recognition, their strengths and weaknesses under three divisions, local, holistic and hybrid methods. The paper highlights important characteristics of any functional facial recognition system as well as the three basic steps in developing one: face detection, feature extraction and face recognition.

Lin (2000) further explains the framework of the face detector and face recognizer, pointing to their shared component, a feature extractor that interprets the pixels of the image of a face as a useful vector, and a pattern recognizer that searches through an existent database to classify the incoming image, either as a face image or non-face image for a face detector or as the face of a specific individual for a face recognizer.

Khairuddin, M. H. *et al* (2021) proposes a technique in which the cloud is used as the primary storage for the information instead of local storage. This ensures that the storage cannot easily be tampered with or destroyed. Dropbox and Cloudinary are both tested with Dropbox having consistently better performance. Haar cascade classifier is used in facial recognition only when raspberry pi picks up movement (motion detection) thereby saving compute. The system is implemented using a Web page and the label 'unknown' is attached to unrecognized personnel.

Milossi, M., (2021) presents the ethical concerns in the application and gathering of the increasingly ubiquitous technology. It also understates the need for regulations and trustworthy regulatory bodies to be set up for ensuring the privacy of the individual.

In the past, the implementation of facial recognition technology was done using traditional computing methods. However, increased research in the field of artificial intelligence has made methods such as machine learning and deep learning widely adopted in performing face recognition tasks. These methods have been proven to successfully outperform other traditional techniques of facial recognition.

Artificial Intelligence: This is a field of science that is focused on building computers and machines with intelligence. This means that they can think, learn and behave in a way which would require human intelligence. It is based primarily on machine learning and deep learning. Machine learning is a subset of AI, in which algorithms make predictions or categorize information based on training data. This data can be labeled or unlabeled, Deep learning uses artificial multi-layer neural networks that mimic the human brain in layout and function. These deep learning models are trained on large amounts of data for a preset number of epochs. The number of epochs for which the models are trained is chosen carefully to avoid overfitting or underfitting Isaac et al. (2024).

The techniques employed for the classification task of facial recognition in the flask application include support vector machine, a machine learning algorithm in the first step and buffalo_l, a model built on the ResNet-50 convolutional neural network and implemented in the InsightFace framework in the second step (Google Cloud, n.d.). Machine learning models have shown demonstrable superiority to other methods of classification, excelling them in accuracy and generalization, as seen in the comparative analysis conducted by Jayaswal and Dixit (2020) where traditional methods achieved an accuracy of 50% while machine learning models achieved accuracy levels of up to 96%.

Although the technology for facial recognition is ubiquitous in present-day society, yet it is still not universally adopted due to various factors such as lack of skill and expertise which are necessary to implement these models and a misconception that implementation of such a system requires vast amount of financial resources.

MATERIALS AND METHODS

Materials

The Layout of this system consists of hardware and software components that function in coordination to provide facial recognition applications.

It consists of an IP webcam which is connected to a Flask application. The flask app contains the machine learning models and web application. It also communicates with a Microsoft SQL (MSSQL) database (SQL Server Management Studio (SSMS)).

Software

The Software component of this study consists of two main elements namely:

Machine Learning Models:

The algorithm of the machine learning facial recognition model is divided into two components:

Support Vector Machine

InsightFace

The above algorithms were chosen for their respective strengths which when used in conjunction resulted in increased efficiency and accuracy.

In a study by Faruqe & Hasan (2009), SVMs were shown to be capable of accuracies up to 98% while the Multi-Layer Perceptron (MLP) had an accuracy of 96%. The MLP's accuracy was only seen to reach substantial quality when at least 50 hidden neurons were present. This posed a problem, as a rise in number of neurons, corresponds to the rise of computational resources such as memory used in the application. From the study, a SVM is ideal for an accurate yet lightweight solution. In addition to SVM's accuracy metrics, its suitability for application with the encodings generated by the face_recognition library, as well as ease of implementation all played a role in its choice as the first step of the face recognition algorithm.

The InsightFace python framework was chosen for its ease of implementation and its accuracy metrics. InsightFace has various specific model packs of which the buffalo_l model was chosen for this project. This model was proven to have an accuracy of up to 90.29% when tested on Africans, this value is of import because all ten individuals involved in the project are Africans (Deep Insight, 2024).

Flask Web Application:

The flask web app can be broadly divided into two sections:

The Frontend

In the design of the front end, HTML, CSS and JavaScript were utilized.

Since the application is a flask application, jinja, a web template engine for Python programming language is used. This template enables the reuse of code without repetition and redundancy. Pages present: DashBoard, Logs, ManageUsers, Live

The Backend

Most of the code for the backend is contained in a Python file named 'app.py'. it contains all the routes to the various pages in the web app. This section of code is also responsible for calling the machine learning models and communicating with the SQL database.

It handles requests between the frontend and the database. It also enters new information into the database and continues to run the facial recognition processes in the background when the application is in use.

Hardware

The hardware components used for this study included a laptop, which served as the server and a phone camera with droidcam installed, which served as a webcam.

The phone used was a Samsung Galaxy A12 with a back camera resolution of 48MP.

Methods

Installations

In order to enable the system carry out the above functionalities, the following libraries and software were downloaded, installed and used:

General Coding: Python programming language, Visual Studio Code, Visual Studio, SQL Server Management Studio (SSMS).

Machine Learning: OpenCV, Face_recognition, Dlib, Scikitlearn, LabelImg.

Web Application: Flask, SQLAlchemy

Gathering images

In the course of this study, the facial images of ten students of the 2023/2024 class of Engineering (final year students) were used. These students were in their early twenties, of Nigerian Nationality and the gender distribution was 3 females and 7 males. Obtaining several images necessary for proper training of deep neural network was accomplished by using python to split videos recorded of said persons into several hundred images. The videos were taken in a well-lit space, during the day time, with a combination of natural and indoor lighting conditions. Several angles of the student's faces were taken in the video as well as their side profiles to create a robust dataset that could account for various possible facial orientations.

Preprocessing images

The method of preprocessing varied depending on the facial recognition approach taken.

SVC

The faces in the images were detected and the encodings of the various faces were generated. They were then saved to a dictionary, which was written to a .pkl file.

InsightFace

In preparing images for training by the insightface framework, a data augmentation step was included. 'imgaug.augmenters' was used for horizontal flip, rotation, scaling, addition of gaussian noise and adjusting contrast of the images.

Train models

The training process for both techniques was also different:

SVC

The '.pkl' file obtained from preprocessing the images was read into a dictionary object. This dictionary was then organized into two separate lists, one containing the values (encodings for each image) and the other containing the labels. A test dataset was also created to ascertain the validity of the classification made by the trained classifier.

It was then saved as a .pkl file.

InsightFace

For the training of the InsightFace model, the augmented images were passed into the insightface model and embeddings for the faces were generated and averaged. This database of embeddings was saved as a pickle file.

Implement models

The trained models were used in the facial recognition process.

First, the process of facial detection is carried out using python's face_recognition library. The SVC classifier is called using a function to run on frames with detected faces. This function takes the frame, face_encodings and face_locations generated by the python's face_recognition. The function uses the classifier to generate an ID for any given face image. If the probability (confidence) given by the classifier is above a given threshold (0.79), the face is saved with that class ID.

If however, the confidence falls below this threshold, the function returns a dictionary with 'None' values in the 'label' key.

After calling this function and saving the returned value to the object 'returned', the value of 'returned['label'][i]' is checked to ascertain the SVC accurately classified the face image. If this value is false (meaning it is empty or has no value), a new list with the facial encodings and locations of the faces that the classifier failed to classify is sent to the 'detect_objects' function. This function which contains the insightface model is then called on the frame that was unrecognized by the SVC. The process is run repeatedly for consecutive frames to obtain the top probability for five consecutive frames. The datetime is checked and saved to the database. The image of the face is also saved with a filename of the form: 'name datetime'.

Unrecognized face encodings are given IDs and saved. These encodings are compared with new unrecognized encodings to detect repeated occurrences of said face.

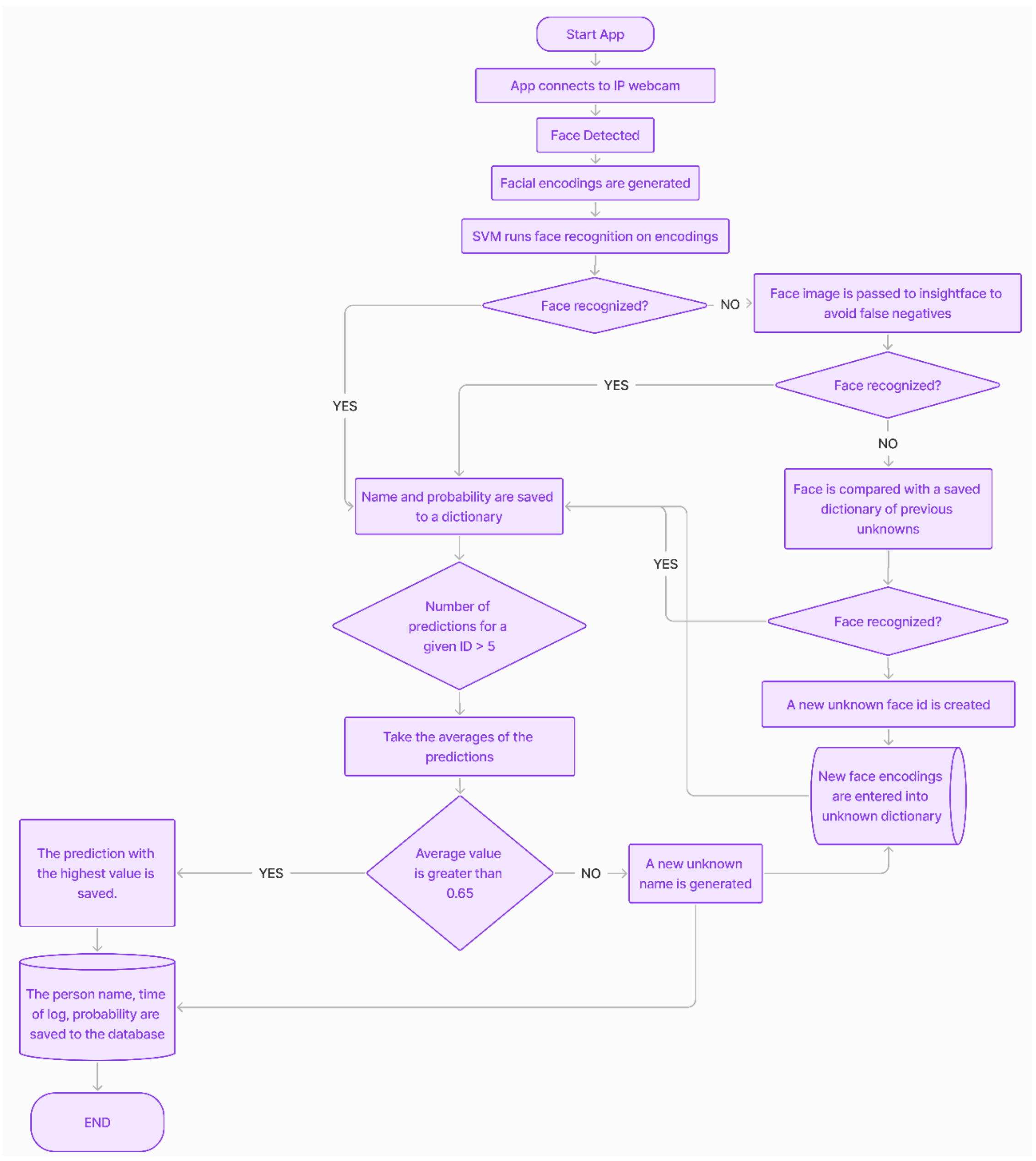

Start App
App connects to IP webcam
Face Detected
Facial encodings are generated
SVM runs face recognition on encodings
Face recognized?
NO
Face image is passed to insightface to avoid false negatives
YES
Face recognized?
YES
NO
Name and probability are saved to a dictionary
Face is compared with a saved dictionary of previous unknowns
YES
Number of predictions for a given ID > 5
Face recognized?
A new unknown face id is created
Take the averages of the predictions
New face encodings are entered into unknown dictionary
The prediction with the highest value is saved.
YES
Average value is greater than 0.65
NO
A new unknown name is generated
The person name, time of log, probability are saved to the database
END

Figure 1: Design plan

Implement frontend

The frontend web design code was implemented, within the static folder and templates folders. The following pages were designed:

Dashboard: The dashboard shows the total number of records in a particular day.

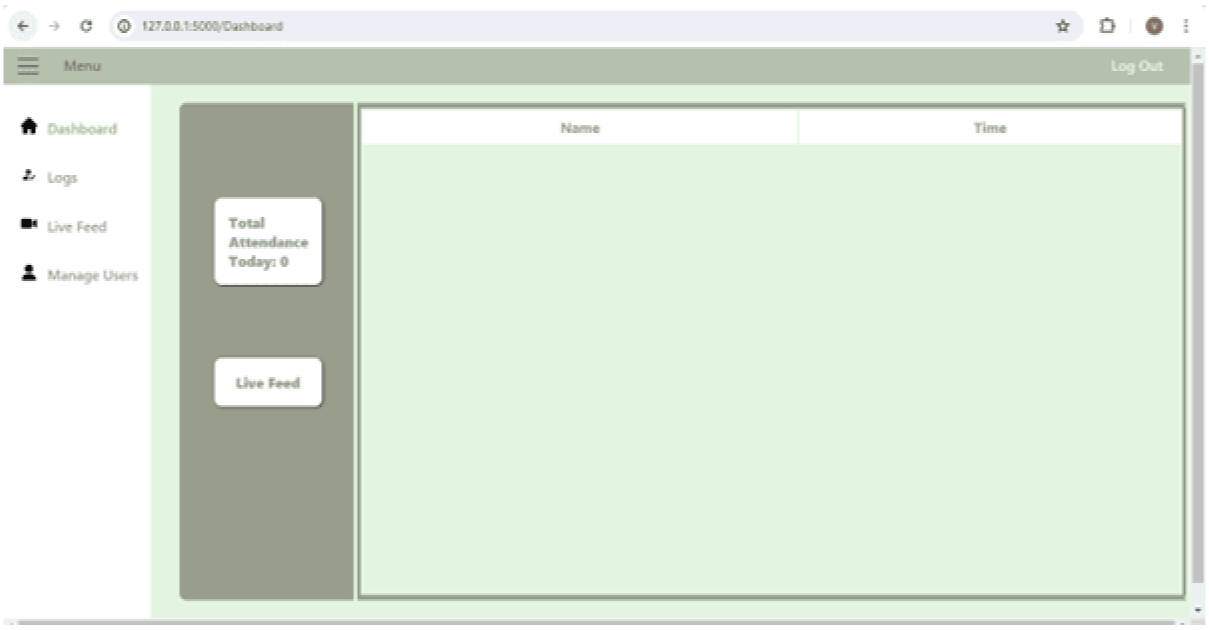
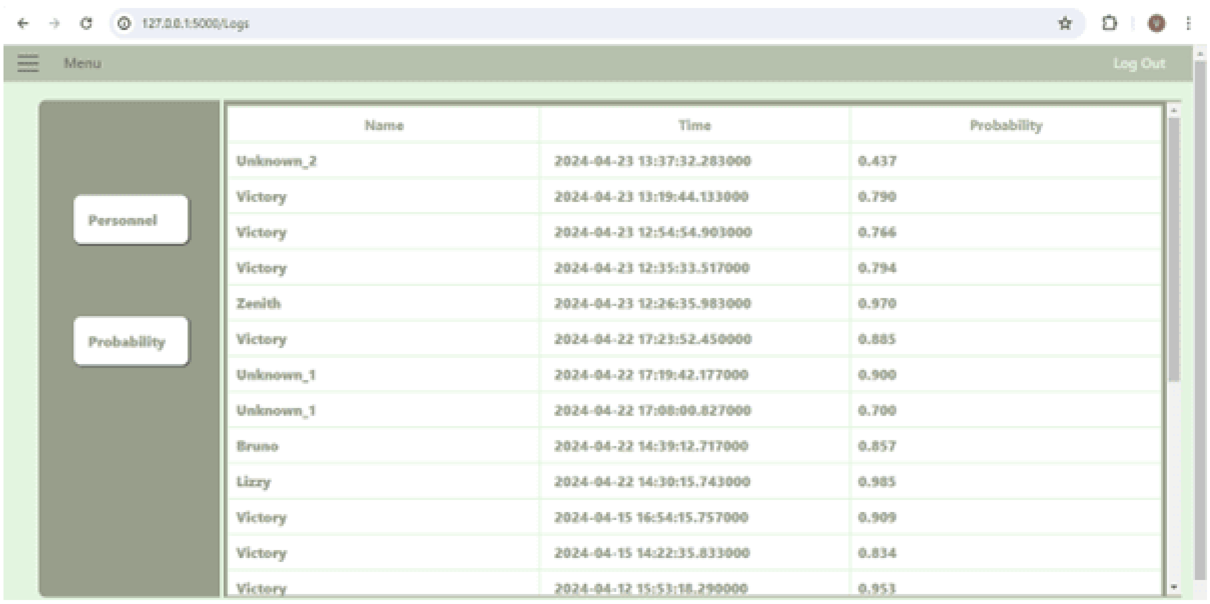


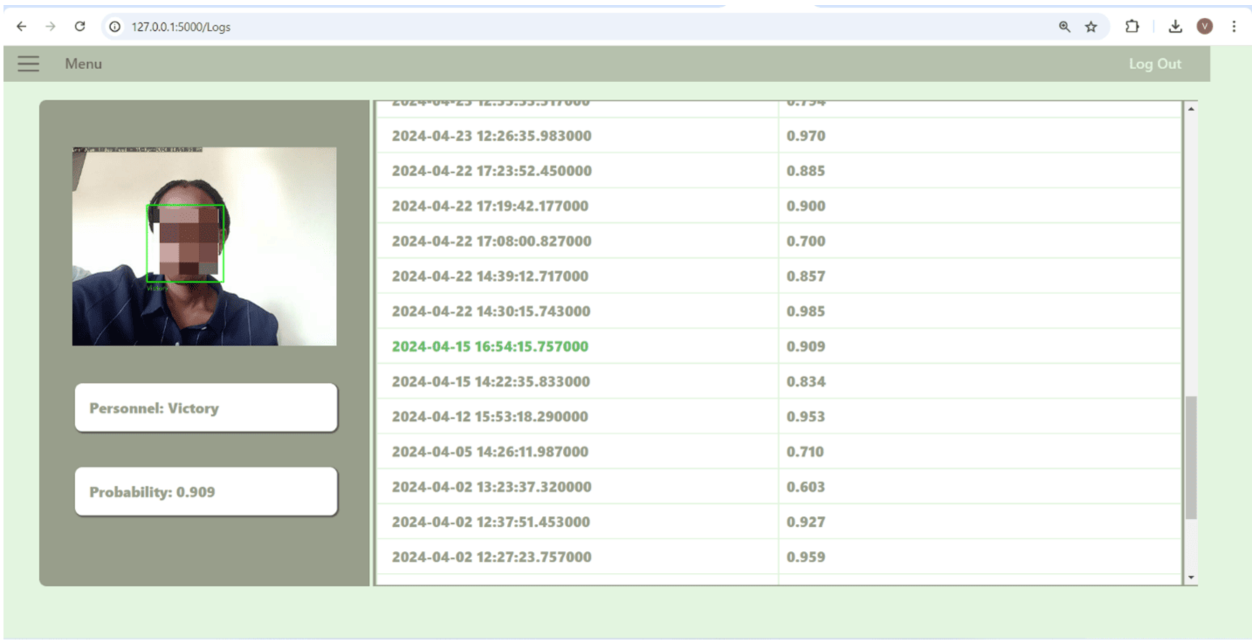
Figure 2: Render of the dashboard

Logs: The logs show the total logs, including datetime which can be selected to view the image of the personnel who was recorded at that time.



Name	Time	Probability
Unknown_2	2024-04-23 13:37:32.283000	0.437
Victory	2024-04-23 13:19:44.133000	0.790
Victory	2024-04-23 12:54:54.903000	0.764
Victory	2024-04-23 12:35:33.517000	0.794
Zenith	2024-04-23 12:26:35.983000	0.970
Victory	2024-04-22 17:23:52.450000	0.885
Unknown_1	2024-04-22 17:19:42.177000	0.900
Unknown_1	2024-04-22 17:08:00.827000	0.700
Bruno	2024-04-22 14:39:12.717000	0.857
Lizzy	2024-04-22 14:30:15.743000	0.985
Victory	2024-04-15 16:54:15.757000	0.909
Victory	2024-04-15 14:22:35.833000	0.834
Victory	2024-04-12 15:53:18.290000	0.953

Figure 3: Render of the logs page



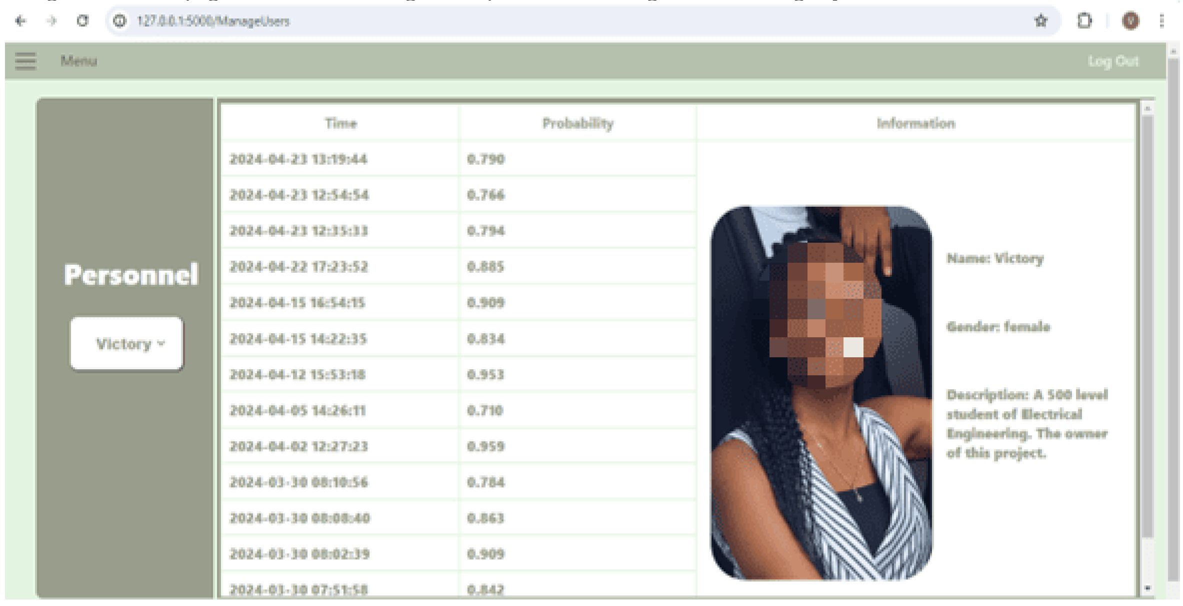
Personnel: Victory

Probability: 0.909

2024-04-23 12:26:35.983000	0.970
2024-04-22 17:23:52.450000	0.885
2024-04-22 17:19:42.177000	0.900
2024-04-22 17:08:00.827000	0.700
2024-04-22 14:39:12.717000	0.857
2024-04-22 14:30:15.743000	0.985
2024-04-15 16:54:15.757000	0.909
2024-04-15 14:22:35.833000	0.834
2024-04-12 15:53:18.290000	0.953
2024-04-05 14:26:11.987000	0.710
2024-04-02 13:23:37.320000	0.603
2024-04-02 12:37:51.453000	0.927
2024-04-02 12:27:23.757000	0.959

Figure 4: Logs page with a time selected

ManageUsers: This page shows the total registered personnel and organizes their logs by ID.



Time	Probability	Information
2024-04-23 13:19:44	0.790	Name: Victory Gender: female Description: A 500 level student of Electrical Engineering. The owner of this project.
2024-04-23 12:54:54	0.766	
2024-04-23 12:35:33	0.794	
2024-04-22 17:23:52	0.885	
2024-04-15 16:54:15	0.909	
2024-04-15 14:22:35	0.834	
2024-04-12 15:53:18	0.953	
2024-04-05 14:26:11	0.710	
2024-04-02 12:27:23	0.959	
2024-03-30 08:10:56	0.784	
2024-03-30 08:08:40	0.863	
2024-03-30 08:02:39	0.909	
2024-03-30 07:51:58	0.842	

Figure 5: Manage users page

Implement Backend

The backend consists of the app.py file which has the routes to the various webpages and sends and receives information from the database.

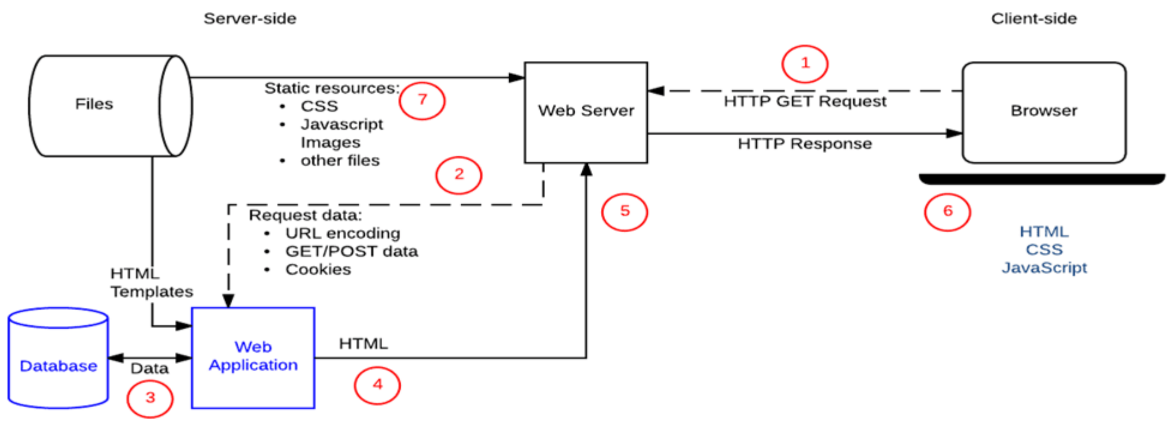


Figure 6: Architecture of server side client side communication
Source: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview

RESULTS AND DISCUSSION

The following are the results of tests on the SVC and insightface models:
The SVM had an accuracy of 90.43%, a precision of 91.20%, a recall of 90.43% and an f1 score of 90.64%. The insightface model's metrics were also acceptable, with an accuracy of 98.78%, a precision of 98.82%, a recall of 98.78% and an f1 score of 98.77%.

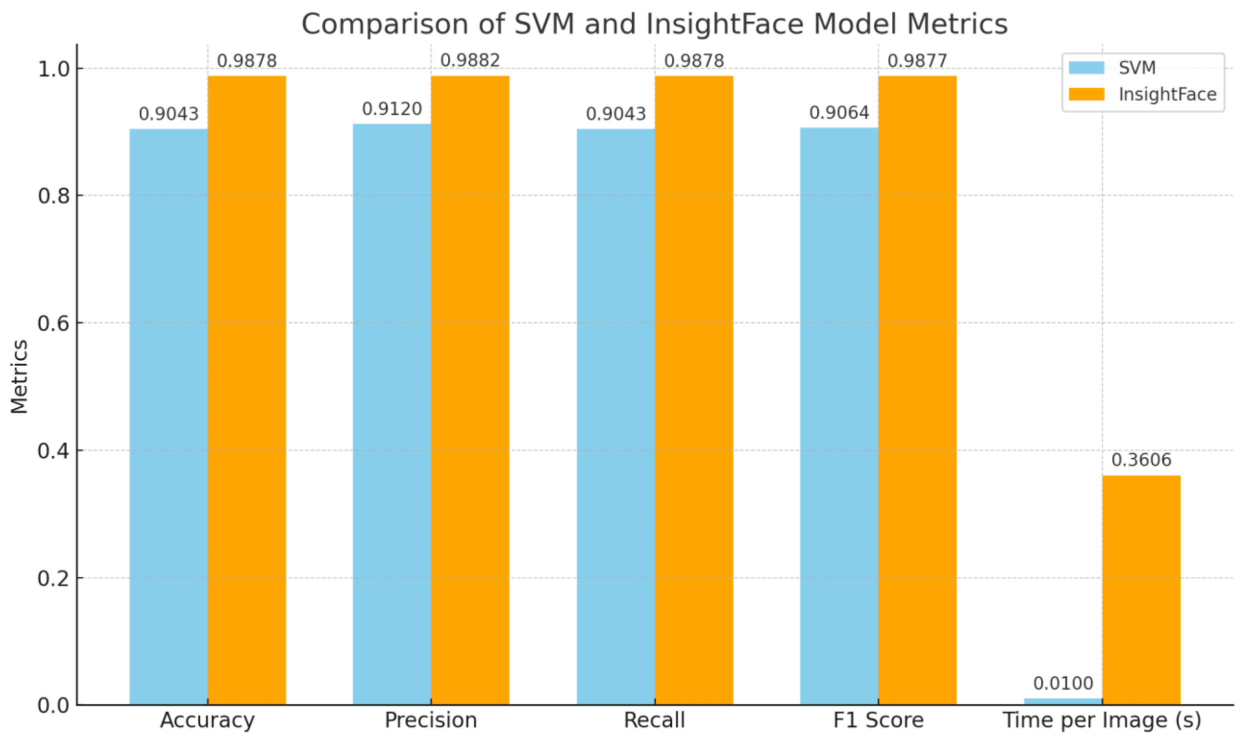


Figure 7: Comparison of SVM and insightface model metrics

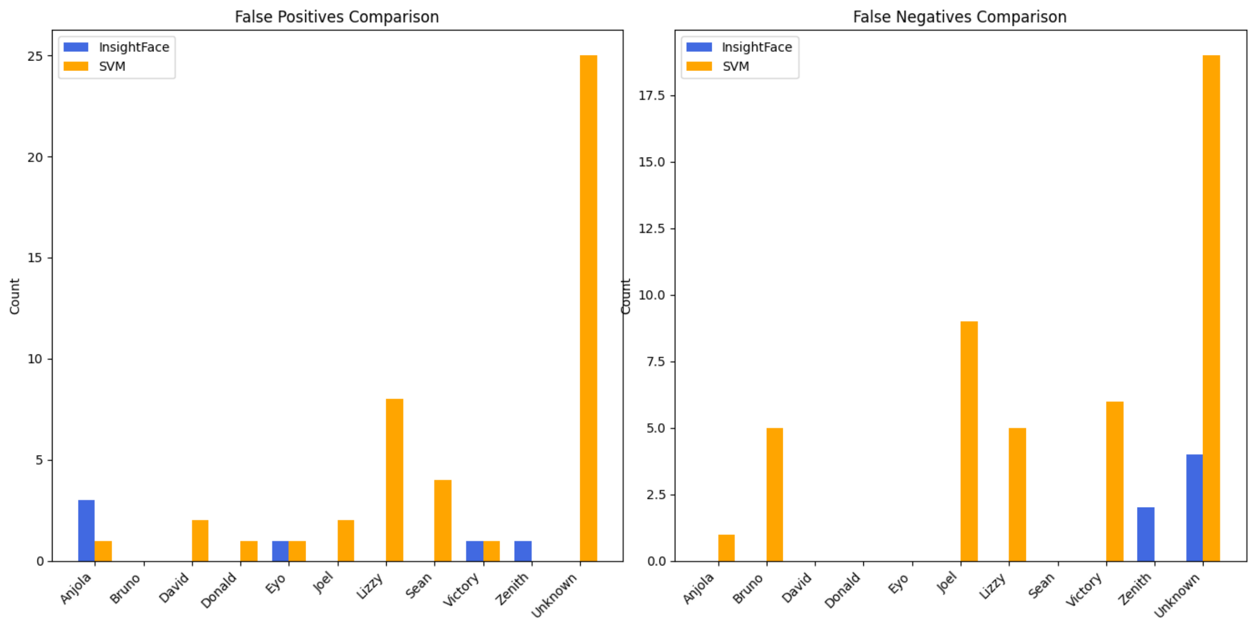


Figure 8: False positives and false negatives comparison for SVM and insightface models

From observation of the heatmap for the SVM classifier, the SVM falls short in accurate classification of unknown persons in the dataset. The threshold chosen to obtain an accuracy of 90.43% was 0.69. However, a higher threshold can be chosen in practice to reduce the number of times the unknown class was misclassified as a person in the database by sacrificing the accuracy. The predictions with probabilities that fall below the threshold are sent to the insightface model which is more accurate. A threshold of 0.15 was chosen for the insightface model to obtain an accuracy of 98.78%.

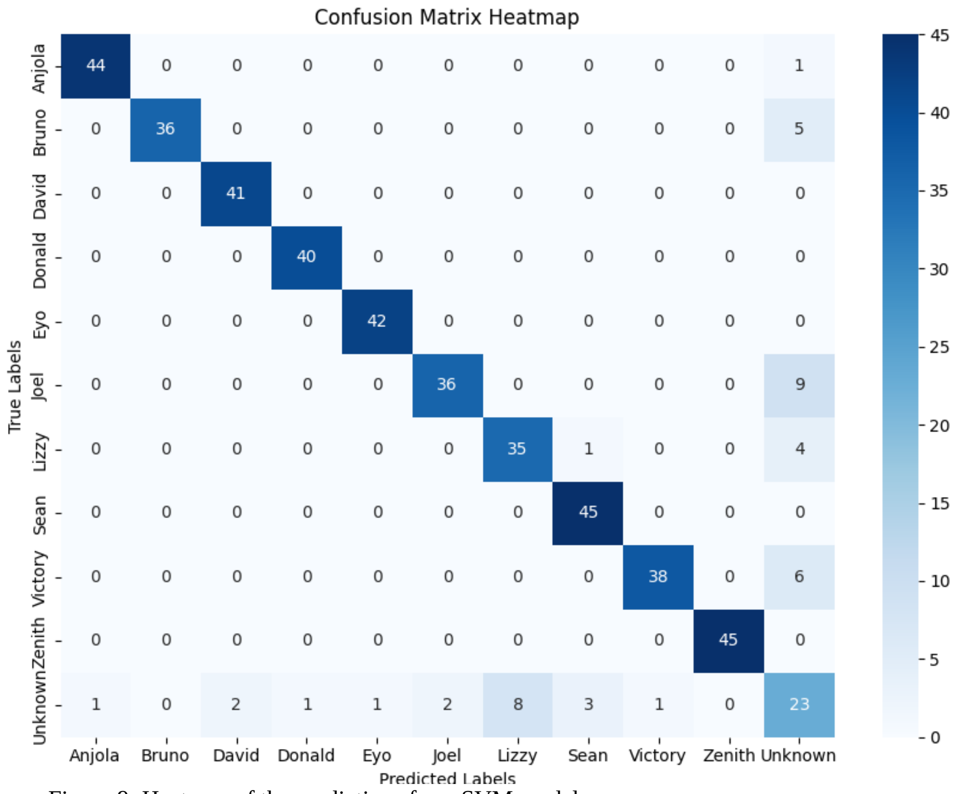


Figure 9: Heatmap of the predictions from SVM model

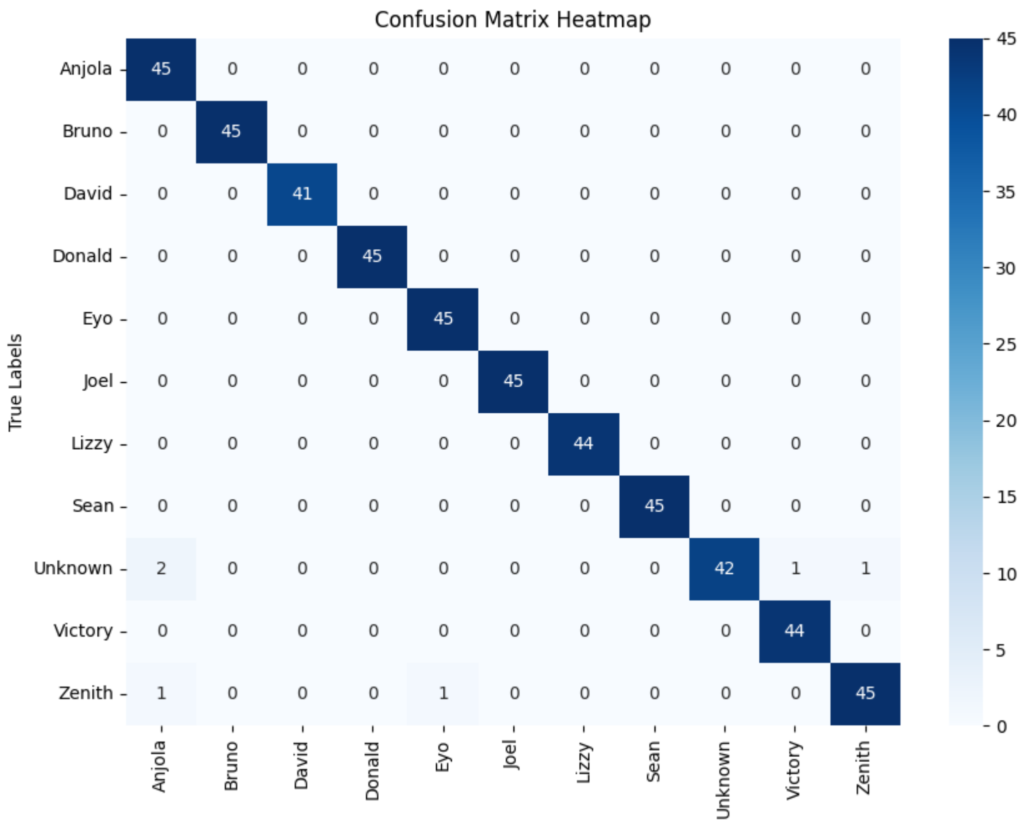
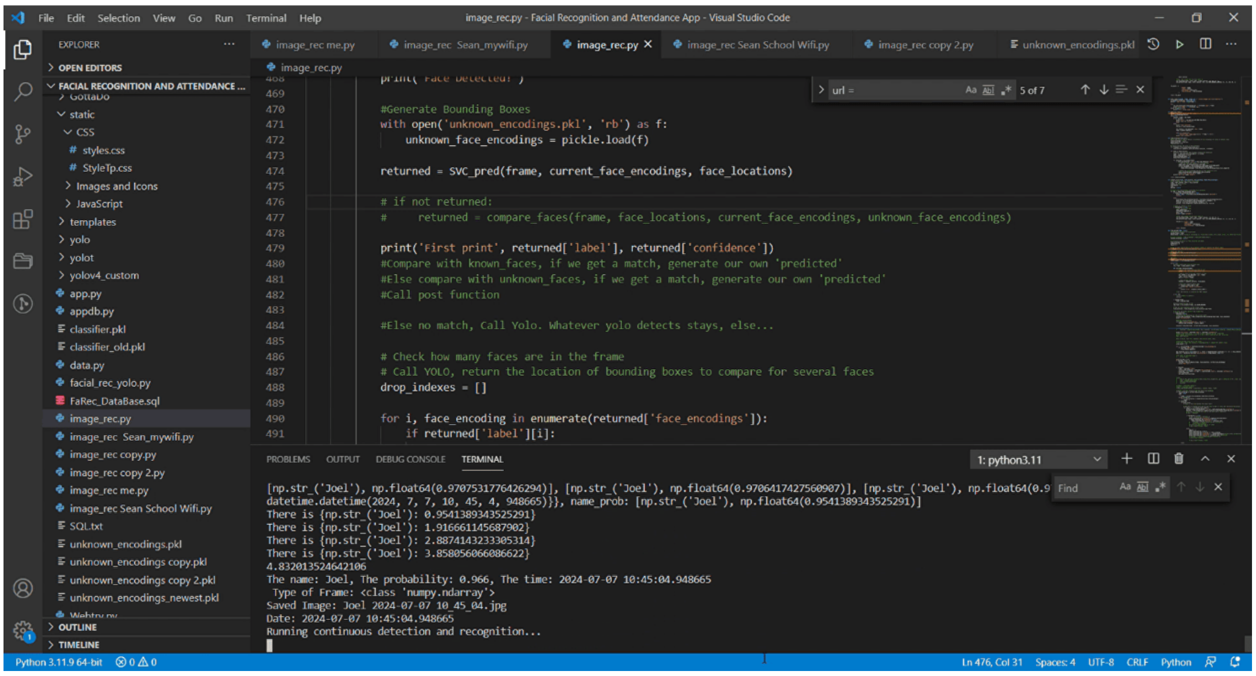


Figure 10: Heatmap of the predictions from insightface model

The system was tested under natural and indoor lighting conditions. The system used for testing was a hp laptop with an intel core i5 processor.
This system can be improved upon by leveraging on data augmentation for the images in the training process of the SVM. In addition, the dataset used includes similar images obtained from a single video, the use of more varied images would improve the performance of the model in real-life scenarios. For instance, some images taken at different times of the day can be added.
Moreover, this dataset includes only ten classes, for a real-world scenario, more classes would be required. The system can also benefit from implementing another, more powerful face detection algorithm to use in tandem with the SVM, insightface remains a viable alternative.
Testing the functionality of the system:



```python
        print('Face Detected!')

        #Generate Bounding Boxes
        with open('unknown_encodings.pkl', 'rb') as f:
            unknown_face_encodings = pickle.load(f)

        returned = SVC_pred(frame, current_face_encodings, face_locations)

        # if not returned:
        #     returned = compare_faces(frame, face_locations, current_face_encodings, unknown_face_encodings)

        print('First print', returned['label'], returned['confidence'])
        #Compare with known_faces, if we get a match, generate our own 'predicted'
        #Else compare with unknown_faces, if we get a match, generate our own 'predicted'
        #Call post function

        #Else no match, Call Yolo. Whatever yolo detects stays, else...

        # Check how many faces are in the frame
        # Call YOLO, return the location of bounding boxes to compare for several faces
        drop_indexes = []

        for i, face_encoding in enumerate(returned['face_encodings']):
            if returned['label'][i]:
```

```
[np.str_('Joel'), np.float64(0.9707531776426294)], [np.str_('Joel'), np.float64(0.9706417427560907)], [np.str_('Joel'), np.float64(0.9
datetime.datetime(2024, 7, 7, 10, 45, 4, 948665)}}, name_prob: [np.str_('Joel'), np.float64(0.9541389343525291)]
There is {np.str_('Joel'): 0.9541389343525291}
There is {np.str_('Joel'): 1.916661145687902}
There is {np.str_('Joel'): 2.8874143233305314}
There is {np.str_('Joel'): 3.858056066086622}
4.832013524642106
The name: Joel, The probability: 0.966, The time: 2024-07-07 10:45:04.948665
 Type of Frame: <class 'numpy.ndarray'>
Saved Image: Joel 2024-07-07 10_45_04.jpg
Date: 2024-07-07 10:45:04.948665
Running continuous detection and recognition...
```

Figure 11: Terminal showing the outputs from running application

The system successfully starts up and runs. It saves the images of unknown and known personnel, displaying them and their times of entry in the terminal, before saving in the system (database).
This information is easily accessible from the application.

CONCLUSION

A facial recognition algorithm was designed using machine learning models, this algorithm was then implemented as part of an attendance application. The models performed with accuracy rates of 90.43% for the SVM and 98.78% for the insightface model and processed a face image with impressive speeds of 0.01s for the SVM and 0.3606s for the insightface model. With these levels of accuracy and processing speed, practical applications such as real-time security and monitoring can be undertaken with the model.
Facial detection is a pivotal part of this study, as this is the tool used to separate useful information such as when a person is approaching from irrelevant movement or objects.
This system would greatly improve the speed of attendance taking especially for large classes or large amounts of people. It is a non-invasive technology and can be deployed to be completely autonomous or to be semi-automatic, with human intervention only at key moments, such as when unrecognized personnel is in the facility.
It can be moved to the cloud, where the administrators would have access to their security data from anywhere in the world and it aids in taking easily accessible and searchable permanent data records.
The areas of application of this technology are limitless, attendance applications just bring one of them. Facial recognition can be used for biometric authentication instead of passcodes or fingerprints for bank transaction, monitoring facial expressions can be used to spot and treat mental, emotional or psychological illnesses such as anxiety or depression.
Although technology this invasive raises several ethical concerns, this technology will only continue to expand, so the best approach is to embrace and harness it rather than shy away from it while the world moves on.

REFERENCES

Adjabi, I., Ouahabi, A., Benzaoui, A. and Taleb-Ahmed, A., (2020) “Past, present, and future of face recognition: A review”. *Electronics*, *9*(8), p.1188.

Barney, N. (n.d.). “Face detection”. *SearchEnterpriseAI*. Retrieved from https://www.techtarget.com/searchenterpriseai/definition/face-detection.

DeepInsight. 2024. InsightFace. GitHub repository. Available at: https://github.com/deepinsight/insightface.

Faruqe, M. O., & Hasan, M. A. M. (2009, August). Face recognition using PCA and SVM. In *2009 3rd international conference on anti-counterfeiting, security, and identification in communication* (pp. 97-101). IEEE.

Google Cloud. (n.d.). *What is artificial intelligence?* Google Cloud. https://cloud.google.com/learn/what-is-artificial-intelligence.

Isaac, S., Ayodeji, D. K., Luqman, Y., Karma, S. M., & Aminu, J. (2024). CYBER SECURITY ATTACK DETECTION MODEL USING SEMI-SUPERVISED LEARNING. *FUDMA JOURNAL OF SCIENCES*, *8*(2), 92-100.

Jayaswal, R., & Dixit, M. (2020, April). Comparative analysis of human face recognition by traditional methods and deep learning in real-time environment. In *2020 IEEE 9th international conference on communication systems and network technologies (CSNT)* (pp. 66-71). IEEE.

Kaspersky. (n.d.). *What is facial recognition?* Kaspersky. Retrieved November 6, 2024, from https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition.

Khairuddin, M. H., Shahbudin, S., & Kassim, M. (2021). “A smart building security system with intelligent face detection and recognition”. *In IOP conference series: Materials science and engineering* (Vol. 1176, No. 1, p. 012030). *IOP Publishing*.

Kortli Y., Jridi M., Al Falou A., & Atri M. (2020). “Face Recognition Systems: A Survey”. *Sensors*, 20(2), 342. https://doi.org/10.3390/s2002034 2.

Lin, S. H. (2000). “An introduction to face recognition technology”. *Informing Sci. Int. J. an Emerg. Transdiscipl.*, 3, 1-7.

Milossi, M. (2021). “Remote biometric identification systems and ethical challenges: The case of facial recognition”. *In 2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)* (pp. 1-6). *IEEE.Paper 12*.

Porter, G., & Doran, G. (2000). “An anatomical and photographic technique for forensic facial identification”. *Forensic Science International*, 114(2), 97-105. https://doi.org/10.1016/S0379-0738(00)00290-5 .