Instituto Tecnológico y de Estudios Superiores de Monterrey

TC3007C-501 Advanced Artificial Intelligence for Data Science II

Class Insight: AI-powered Software for Automation of Attendance and Participation of Students in the Classroom

Jorge Eduardo De León Reyna - A00829759
David Esquer Ramos - A01114940
Francisco Mestizo Hernández - A01731549
Adrián Emmanuel Faz Mercado - A01570770

November, 2023

Class Insight: AI-powered Software for Automation of Attendance and Participation of Students in the Classroom

Jorge Eduardo De León Reyna, David Esquer Ramos, Francisco Mestizo Hernández, Adrián Emmanuel Faz Mercado

November, 2023

Abstract

Computer vision has multiple applications and it's widely used to do face and pose recognition. This can be accomplished using deep learning algorithms for image analysis. For example, Python's face-detection library from MIT, YOLOv3 and mediapipe. The project consists of an advanced system for automated attendance registration and detection of student participation in the classroom. This system provides teachers with a tool to efficiently manage and automate the tracking of student attendance and participation. Additionally, it is complemented by an integrated platform that facilitates the visualization of statistics and effective management of courses and students. The system is efficient and easy to use. It works perfectly in controlled conditions, but it could fail to detect a face or body position in certain conditions such as dim rooms, people too far away from the camera, and the use of accessories like hats or glasses. Overall the face detection model has 86% of accuracy and the pose detection mode has a 74% accuracy.

1 Introduction

In the current educational landscape, one of the most significant challenges is the efficient management of time in the classroom. Teachers often invest a considerable amount of their time in administrative tasks such as taking attendance and tracking student participation, instead of focusing on teaching and interacting with students.

In this context, the need arises for a solution that can simplify and automate these administrative tasks, enabling teachers to dedicate more attention and resources to what truly matters: teaching and motivating their students.

Using this kind of software can generate problems. Even though they are developed using AI, they can still be fooled. For example, using a picture of an absent student to mark

the attendance. Another problem could be the size of the classroom or amount of students. If there is a room where the students are too far, does not have the right lighting or the hardware does not have the necessary resolution, it is more probable that the system fails.

In the rest of the document there will be an explanation of the key points of the process that was done to get to the final MVP. Going through the technologies used in Section 2, the development in section 3, testing in section 4 and results in section 5.

2 Fundamentals

There are several investigations to do implementations of this in real classrooms. According to [1] and [3], the most efficient ways to do face recognition are Convolutional Neural Networks. On the other hand, [2] and [4] say that there are several methods to do human pose recognition, for example using YOLO or using the pH36M method and Rapid Entire Body Assessment (REBA).

Most of the technology used for this project is the state of art of artificial intelligence, that is why some key points will be described in the section. From general theory concepts to the description of each algorithm that is used in each module.

2.1. Computer Vision

Computer vision is a rapidly advancing field within artificial intelligence that focuses on enabling computers and systems to interpret and understand visual data from the world around them. This branch of artificial intelligence uses digital images and videos to emulate human visual perception, allowing machines to recognize, analyze, and make decisions based on visual input. This field combines techniques from AI and machine learning, aiming to replicate the complex process of human sight in machines. As a result, these systems can perform tasks that require visual comprehension, such as identifying objects or interpreting scenes. [5]

This field has evolved to become an essential component of many modern technologies. It plays a vital role in different applications, ranging from everyday tasks to complex industrial systems. As an example, in the healthcare area, computer vision helps in analyzing medical imagery for better diagnosis and treatment plans. In the automotive industry, it is a key technology behind self-driving cars, enabling these vehicles to navigate safely by understanding road conditions and surroundings. As well, in the entertainment

sector, computer vision enhances user experience in video games and virtual reality, providing more immersive and interactive environments[6].

The advancement of computer vision has been rapid through the last years. Its accuracy and capabilities have significantly improved due to the development of new algorithms and processing techniques. Some of these algorithms rely on deep learning models such as Recurrent Neural Networks (RNNs), Fully Connected Networks (FCNs) and Convolutional Neural Networks (CNNs). These models help to process images, identify patterns and distinguish features, enabling the segmentation of complex visual scenes into understandable components. This segmentation is crucial for tasks such as object recognition, facial identification, and scene understanding, making these models indispensable for a wide range of applications

2.2. Face Recognition

Face recognition and face detection are essential aspects of contemporary computer vision systems, revolutionizing how machines interpret human features. The term “face detection” involves identifying human faces with digital images, and distinguishing them from the background and other objects. This process has been significantly enhanced by advanced algorithms, such as the Haar cascades and different deep learning techniques, which have improved the detection accuracy under diverse conditions, including variations in lightning, angles and facial expressions. [7]

Upon face detection, face recognition technology takes the task further by verifying individual identities. This process analyzes unique facial features and landmarks using deep learning models to learn and process big datasets of facial images. These models produce facial embeddings, which are numerical representations of facial features, enabling accurate identity matches.

Generally, face recognizers try to find essential feature points such as eyebrows, corners of the mouth, lips, and others. These points, often exceeding 60 in number, serve as a map of the face's unique features. By accurately detecting these landmarks, the system can analyze a face's structure, expressions, and orientation, which are essential for various tasks in computer vision. [8]

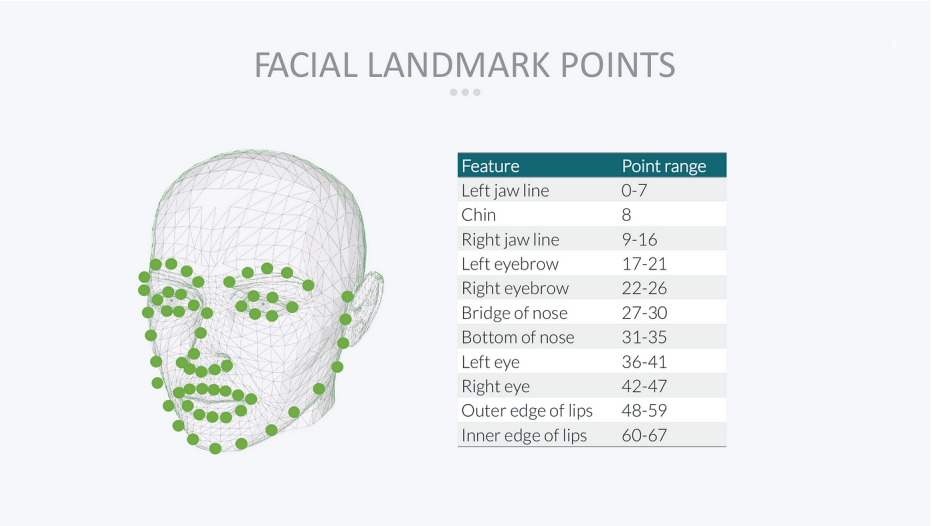

Figure 1. Feature points for face recognition. - Source

The process that is usually performed when recognizing a new face begins by locating the face, detecting the coordinates of each face, and drawing a bounding box around every face. Then, once a face is detected, it identifies the specific key points or landmarks on the face. After, the system extracts facial features, including the distance between the eyes, shape of the nose, contour of the cheekbones, and is achieved with different deep learning methods. The embeddings are then compared against a database of known faces, and this is done by methods such as making simple distance measurements in feature space or more complex matching algorithms.

There are different existing algorithms for facial recognition, each employing different methods to accurately try to identify human faces:

- **FaceNet:** Algorithm developed by Google that uses Convolutional Neural Networks. It maps facial images into an Euclidean space where the distance between images indicates their similarity.
- **ArcFace:** It focuses on creating a clear separation between predefined classes for face verification and identification. Employs similarity learning with angular margin loss to enhance class separation. As well, it optimizes angles between class embeddings and class weights for different facial feature differentiation. [9]
- **DeepFace**: Algorithm developed by Facebook that uses deep neural network to recognize faces in images and videos with high accuracy. It uses a multi-layer neural network to map facial features into a vector space where similarity between faces can be measured.

- **Eigenfaces:** Algorithm based on principal component analysis (PCA) and focuses on dimensionality reduction. It represents the principal components of variation in a dataset of face images. By projecting a new face image into the “eigenface” space, it can be reconstructed using a small number of coefficients, which effectively compresses the facial features. Then, recognition is performed by comparing those coefficients.

2.3. Pose Estimation

Pose Estimation is a crucial area in the field of Artificial Intelligence and Machine Learning. “Pose estimation” involves the process of detecting the position and orientation of an object or a person, focusing mainly on identifying key points of the human body, such as elbows, wrists, knees, ankles, and hips. This technology interprets the spatial configuration of a human or an object in images or videos, which is often represented through a skeleton model. In this way, when an image or video is given to the pose estimator model as input, it identifies the coordinates of specific body parts and joints as output, and a confidence score showing the precision of the estimators. [9]

In normal object detection, people are simply perceived as a bounding box once they are identified as a person. However, by performing pose detection and estimation, computers can have a better understanding of human body language. These advancements have a significant importance in different sectors. For example, in healthcare, pose estimation facilitates advanced patient monitoring and physical therapy, allowing for precise tracking of rehabilitation progress. In sports, coaches, and athletes use this technology to analyze athletic performance, identifying nuances in posture and movement that could lead to improved techniques or reduced injury risk. This technology opens new doors for more intuitive and natural interfaces, where computers can understand and respond to human gestures and postures. [10]

2.3.1 Types of Pose Estimation

Pose estimation can be categorized into 2 different types: 2D and 3D pose estimation. Each type offers different capabilities and finds applications in different domains:

- 2D Pose Estimation: Type of pose estimation used to estimate the 2D position or spatial location of human body key points from images or videos. The location is represented with X and Y coordinates for each key point.
- 3D Pose Estimation: Type of pose estimation in which a 2D image is transformed into a 3D object by estimating an additional Z-dimension to the prediction. It allows one to predict the specific spatial positioning of a person.

2.3.2 Types of Pose Estimation Models

To represent the human body in 2D and 3D planes, there are three main types of human pose estimation models:

1. Kinematic model: It uses key points (joints) such as ankles, knees, shoulders, elbows, and wrists for both 3D and 2D pose estimation. It represents the skeletal structure of the human body and it is normally used to understand the relationships between different body parts.
2. Contour-based Model: It includes the body's contour and approximate width. It represents the human body's appearance and shape, displaying body parts with boundaries and rectangles that outline a person's contour.
3. Volume-based Model: It consists of multiple 3D human body models and poses represented by geometric meshes and shapes. Used specifically for 3D pose estimation.

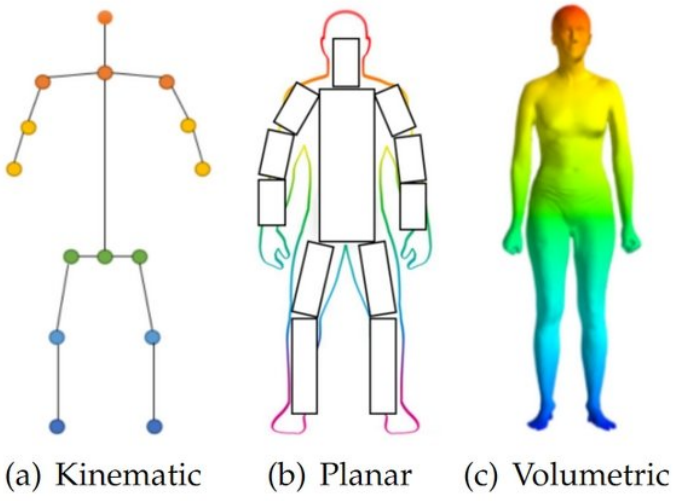

Figure 2. The three types of models for human body modeling - Source

2.4 Selected Algorithm for Face Recognition

In this project, the *face_recognition* library from Python was used for face detection and face recognition. This library combines classical methods with modern deep learning techniques to provide accurate and real-time facial analysis.

Face Detection

To detect faces, the program employs the **Histogram of Oriented Gradients (HOG)** as the primary algorithm for face detection, a robust feature descriptor widely used in computer vision. This algorithm breaks down an image into small, manageable regions known as cells. Within each of these cells, it computes histograms based on the directions of pixel gradients, effectively capturing the structural shapes that indicate facial features.

Then, after extracting those HOG features, they are analyzed using a linear **Support Vector Machine (SVM)** classifier. This classifier is great at differentiating between areas containing a face and those that do not, scanning through the image and evaluating each section for the presence of facial characteristics. To enhance the accuracy of face detection, post-processing steps like non-maximum suppression are implemented. This technique refines the detected faces, ensuring that each face in the image is identified accurately and uniquely, thus preventing multiple detections of the same face.

Face Recognition

For face recognition in our project, we use a deep learning model based on the Convolutional Neural Network (CNN) architecture, specifically a modified version of the ResNet-34. CNNs are known for their efficacy in image recognition tasks, mostly because of their unique ability to automatically and adaptively learn spatial hierarchies of features from input images.

One of the most important parts of the face recognition process is face embedding. The CNN used processes each detected face to generate a 128-dimensional feature vector. This vector is a numerical representation of the face's distinct features, unique to each individual. The embedding process allows the system to differentiate between various people with good precision. For the training phase, the face-recognition library uses the ResNet-34 architecture, which is trained on a vast and diverse dataset of facial images. This training

equips the model with the ability to recognize a brad spectrum of facial features and nuances, enabling it to encode these characteristics accurately into the embeddings.

To identify or verify a face, the system measures the similarity between the embedding of a detected face and the pre-established embeddings of known faces. This is done by calculating the Euclidean distance between the embeddings. The proximity of these vectors serves as an indicator of identity, in which a smaller distance suggests a higher probability of the embeddings belonging to the same person. To ensure reliability in recognition, a tolerance threshold is used, and if the computed distance between the embeddings falls below it, the system can confidently identify the face as the known person. However, if the distance is greater than the threshold, the system will classify the detected face as an unknown individual.

2.5 Selected Algorithms for Pose Detection and Estimation

Pose Estimation is essential in this project, mostly to detect when a student raised their hand by detecting the position of different body parts. To begin detecting a student's pose, the project uses **YOLOv3**, which is a deep learning algorithm widely recognized for its effectiveness in real time object detection. YOLOv3 uses a Darknet-53 architecture, which is a neural network with 53 convolutional layers for feature extraction. The algorithm uses anchor boxes to recognize various object shapes and sizes. In this system, YOLOv3's ability to simultaneously detect multiple individual is crucial, because this enables the system to work efficiently in classroom environments where multiple students are present. In general words, the algorithm processes video frames to identify subjects for pose estimation.

Upon successful detection of individuals in the frame, **MediaPipe** is used for pose estimation. MediaPipe is an open-source framework that offers tools for real-time pose tracking. MediaPipe uses a combination of convolutional neural networks to estimate the key body landmarks. It currently identifies 33 2D landmarks on the human body, and uses these landmarks to infer the pose and orientation of the body. In the project, MediaPipe takes over

once YOLOv3 identifies people in the video feed. It analyzes each detected person to estimate their pose, focusing on key points like the position of arms and hands.

The final step in the pose estimation process is to interpret the pose data for gesture recognition, to identify hand raises, which mean that a student is participating. To do this, the system analyzes the relative positions of shoulders, elbows and wrists. A hand raise is detected if the wrist is above the elbow, and the elbow is above the shoulder.

3 Methodology and Proposed Solution

In this section, it is described the architecture, the system development and the evaluation metrics. The architecture explains how each part of the system is connected with the others and how the data is shared and accessed for each one.

3.1. Selected method

To determine all the functionalities the software should have, the team had several sessions with the client. Before they began the project, they wrote a requirements document that was shown to the client to clarify what the software would and would not do. Also, a Memorandum of Understanding (MOU) was written to clarify the compromises for each part involved in the development of the project. For more details of the requirements document go to appendix 1 and for the MOU go to appendix 2.

The MVP was developed in less than 10 weeks. As there was too little time, it is important to address the problems in the development quickly. Also, there must be a lot of communication between all the team members. So, SCRUM methodology was used, where the team members had a meeting each two days to showcase their progress. And each Monday the progress was shown to the client to get feedback and rework what was necessary. To learn in detail each sprint go to appendix 3 and for the communication evidence with the client go to appendix 4.

After the project was completely planned, the architecture was established. It has 3 different modules that will be explained further in this section. Roughly, the system consists of the Nextjs project that has the whole frontend of the application. Inside it, there are API endpoints that connect all the other modules to the database through the Internet. Then, the

face recognition module and pose detection module are both developed in Python. Both run locally on a computer and use the camera of the device to get the information.

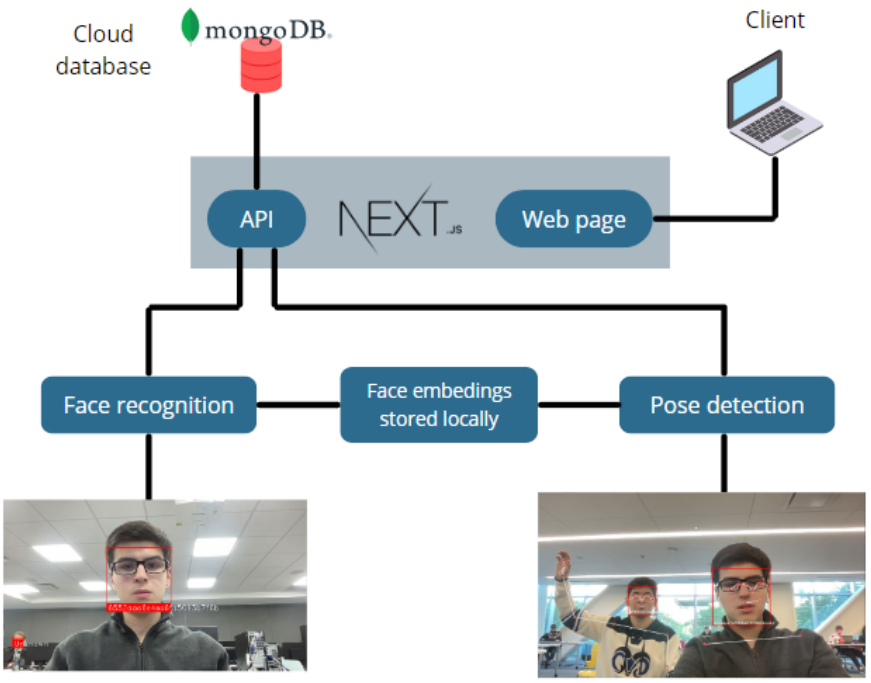

Figure 3. Architecture for the system.

There were several approaches to this use case. The first could be to develop a model from scratch to do image recognition. The problem is that training a model requires a lot of computational resources, time, and a big dataset. This applies to both facial recognition and pose detection models.

Another approach could be to use pre-trained models. This is feasible because implementing a model is easy, you only need to download the weights and use them in a CNN with the same architecture.

3.2. Solution System

For easy implementation, Python's face-detection library from MIT is used. This library includes several methods to detect faces, without the need of downloading the weights. When this algorithm detects a new face, it can generate an embedding from that face and store it. This is beneficial as the embedding can't be turned back to the original face, so the user's privacy is protected. Also, to train the model to detect each person, only one photo is taken, so increasing the number of photos to generate each embedding can improve the model accuracy.

For pose recognition, YOLOv3 and mediapipe were used. As this is a more complex task, it has more steps. First, with YOLOv3 the algorithm detects if there is a person in the frame. After it is detected, mediapipe generates points on the body (face, shoulders, arms, torso and legs) to track the movement of the person. Finally, when it detects that the point in any shoulder is below the point of an elbow and at the same time the point of the elbow is below the point in the hand, a participation is marked.

And for this to be a functional software, it is necessary to have a way to access and analyze data. That's why this system has a web page where dashboards are generated. It has two different roles, the student and the administrator. Students can watch their latest attendances in each of their courses and can see the participations for each course on a calendar to see how often they participate in class. Also, administrators can see all the courses, access them, and see the attendance for each student and the accumulation of attendance and participation by day.

The information collected by the AI is connected to the database hosted in MongoDB and the webpage through an API. These endpoints were created directly in the same Nextjs webpage project. This way, the platform is created using microservices and can easily be maintained.

Privacy is one of the most important considerations for this project, given that facial recognition, powered by artificial intelligence, remains a highly controversial and sensitive topic in today's society. While the technology offers remarkable capabilities, including enhanced security and seamless user experiences, it also raises significant concerns about personal privacy and civil liberties.

All stored data is secured in the database. For facial recognition, as mentioned above, the system only stores the embeddings of each face, which can't be converted back to an image of the face. Additionally, each embedding is stored with the id of the user, not his name. This means there is no way to identify the student outside of the server that hosts the database.

3.3 Data usage

For now, the project does not plan to use big data, as the purpose is solely to create a proof of concept and demonstrate whether it is a viable project to develop or not. However, in

the case of developing this project as a final product, the use of large amounts of data should be considered.

Firstly, the fact that all students, courses, and teachers are included makes it necessary to store information in the database. To enable easy scalability, MongoDB is used as the database, as described above. On the other hand, when working with artificial intelligence models that will be running on external devices, they must have access to a database where it is not as important how the data is stored, but rather that they just store it (non-relational).

The system is based on generating dashboards for the students and courses. So in future implementations, there will be the need to implement big data to handle and process the data and generate more complex dashboards. As there is a lot of information generated on each classroom, Spark can be a really useful tool to process batches of data and display the information on the webpage.

3.4. Metrics for evaluation

According to the bibliography named above, it was decided to test the model using accuracy since it is the most commonly used metric. This means the model should always be able to correctly detect each face and participation, because the measurements could be used for grading purposes.

4 Results

To test the model, metrics such as F1 score, recall, precision, Average Percentage Error (APE) and Mean Squared Error (MSE) will be evaluated. To calculate these metrics, it is necessary to define the four possible outcomes obtained from the models. First, False Positives (FP), when the result should be negative, and it predicts correctly; False Negatives (FN), when the result should be negative but it gives a positive result; True Positives (TP), when the result should be positive, and it predicts correctly; and True Negatives (TN), when the result should be positive but it gives a negative result. To go deeper in the calculations of the metrics of both models, go to appendix 5.

4.1. Face Recognition model testing

The accuracy on the testing for face detection was 86%. Different poses and positions were tested, using three different students. For each one, they took a photo near the camera,

one meter away from the camera, at the side of the frame or with half face outside and one semi profile photo.

```
Accuracy: 85.71428571428571
```

Figure 4. Accuracy for face detection

Most individuals in the tests were identified correctly, and further improvement can be achieved by reducing sensitivity to avoid confusing similar faces. The face detection can fail when there is dim light, there is a strong light from behind, or if the student is wearing glasses or a cap.

4.2. Pose detection model testing

The tests that can be performed on the model involve recording videos with different numbers of people, each either participating (raising their hand) or not. To evaluate performance, these videos will be processed by the system, and FP, FN, TP and TN will be recorded. In this system, FP is considered when the system fails to detect participation if no one raised their hand; FN is when the system detects participation where there is none, either due to incorrectly detecting the pose or confusing elements in the environment; TP will be recorded if it detects participation when a person raised their hand; and TN is when students move but do not raise their hand, and the model do not record a participation. The following table shows the test set and the results obtained:

Description	**Real participations**	**Detected participations**	**TP**	**TN**	**FP**	**FN**
A person close but out of frame.	3	5	3	0	2	0
A person nearby with hands semi-raised but not participating, for example, touching their face.	3	4	3	2	1	0
A person doing nothing.	0	0	0	2	0	0
A video without people.	0	0	0	2	0	0
Two people, one close and one far. Some participations are simultaneous.	5 (3 y 2)	7 (4 y 3)	5	0	2	0

Two people far away. One person keeps their hand raised for a long time.	5 (3 y 2)	6 (3 y 3)	5	0	1	0
One person with another person walking in the scene.	3	6	3	0	2	1

```
Precision: 0.7037037037037037
Recall: 0.95
F1 Score: 0.8085106382978724
Accuracy: 0.7352941176470589
```

Figure 5. Metrics calculated for the participation model

The results obtained indicate that the model does not perform as well in general cases but tends to have some errors. 74% of the participations were correctly detected, and the rest were not. It's important to mention that these measurements were made by comparing whether the total participations in the videos were correct or not.

The root of the error will be analyzed using the following figure. The plot shows each video that was part of the testing dataset. They are labeled from 1 to 7. To see the details of each video, refer to the table above. The blue plot shows the amount of participations (hands raised) were on the video. This does not take into account the amount of people in the video, only the participation that they all sum. The orange bar represents the participations the model detected.

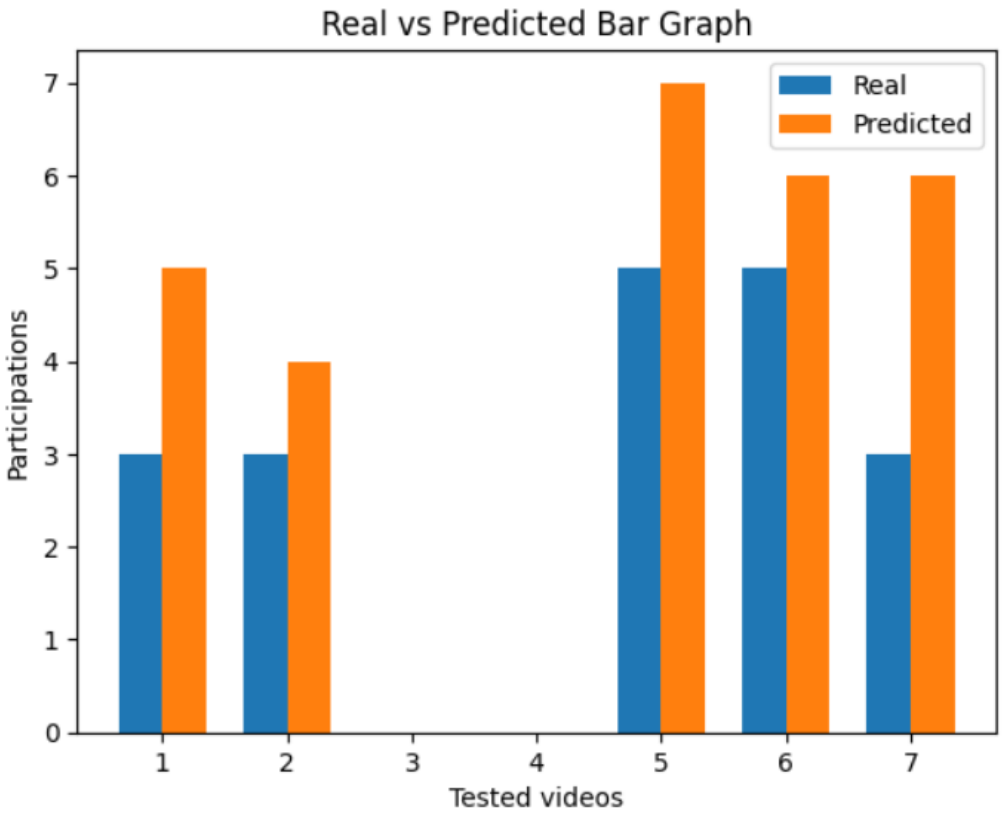

Figure 6. Pose detection testing results

The plot indicates the number of participations that should have been and those that the model counted. Therefore, it is evident that the model is always making predictions above what was labeled. But, for videos where there was no participation, it got them right. This could mean the model is taking double participation, but not getting confused by movement or other elements on the scene.

At the same time the accuracy was calculated, the code measured the error the predictions had. The model makes accurate predictions for student participation and does not undercount them. However, it seems to be very sensitive because it tends to detect participation where there is not. This behavior is evident even when marking more than one participation, even if the hand was raised only once.

With the APE and MSE it is seen that 37% of the participations are detected incorrectly and there are 2.7 more participations detected than what should be.

```
Mean Squared Error: 2.7142857142857144
```

Figure 7. MSE of the model

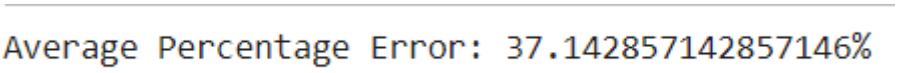

Figure 8. APE of the model

This result indicates the model needs to become more precise in detecting participation and avoid false positives. This can be achieved by changing the way hand-raising detection is performed.

4.3. Integration testing

For the system to work correctly, it is important that all the modules can communicate with each other and share information. They are all connected by the Nextjs API endpoints. After each endpoint was enabled, it was tested with Postman using fake data to verify that the information was delivered correctly. After that, those endpoints were implemented on each Python script to send the information to the database. As the face recognition is in real time, all the data is sent instantly. On the other hand, participation detection is asynchronous. This means that the video of the class is recorded and processed afterwards. And after the processing all the data is sent to the database. All the collected data is then fetch by the web application and displayed in the dashboards.

5 Conclusions

This software is an integration of artificial intelligence pretrained models to do face recognition for taking automatic attendance and pose detection to count the participation of students in class. All these tools are connected and display the information in dashboards for students and teachers on a web application.

The face detection model has an accuracy of 86% and the pose detection model has an accuracy of 74%. This means the results that are shown in the dashboards by class are pretty accurate. The face detection model works fine using a few photos of the student to generate the embedding and detect him at a considerable distance. And the pose detection model works fine even though there are more than three people interacting or participating at the same time. There can be people walking behind or scratching their faces and the model detects correctly that they are not participating.

Using open source libraries can lean the models to not be accurate on some edge cases. Face detection fails to detect people using accessories on their faces, if they are too far

away or if they turn around. This can be improved with better quality cameras and training the model with more photos of each student to generate more accurate embeddings. Pose detection gets confused when there is a source of light behind the students, and can detect participations on it even though no one raises their hands. Also, if the students are too close to each other, the model can count extra participations for students that did not participate. This can be solved by using a more accurate model to track each part of the body and using a more complex logic to detect a participation, not just the points of the arm.

Bibliography

[1] Bhargavi, P. et al. (2022). *Face Recognition Attendance System for Online Classes*. Sardar Patel Institute of Technology: India. https://www.scopus.com/record/display.uri?origin=recordpage&zone=relatedDocuments&eid=2-s2.0-85146342954&noHighlight=false&sort=plf-f&src=s&sid=a87d2b9be098d82bb01af4345663e0e4&sot=b&sdt=b&sl=47&s=TITLE-ABS-KEY%28attendance+taking+neural+network%29&relpos=0

[2] Budiman, A. et al. (2023). *Student attendance with face recognition (LBPH or CNN): Systematic literature review.* Bina Nusantara University: Indonesia. https://www.sciencedirect.com/science/article/pii/S187705092202186X?ref=pdf_download&fr=RR-2&rr=8264c5179d6749e9

[3] Hung-Cuong, N. (2023). *YOLO Series for Human Hand Action Detection and Classification from Egocentric Videos*. Czestochowa University of Technology: Poland. https://www.scopus.com/record/display.uri?eid=2-s2.0-85151215953&origin=resultslist&sort=plf-f&src=s&sid=37590ee2bbeec52f39fc5ddb825398f5&sot=b&sdt=b&s=TITLE-ABS-KEY%28human+pose+AND+detection+AND+yolo%29&sl=29&sessionSearchId=37590ee2bbeec52f39fc5ddb825398f5

[4] Zewei, D. et al. (2023). *An attention-based CNN for automatic whole-body postural assessment*. University of Wollongong: Australia.https://www.scopus.com/record/display.uri?eid=2-s2.0-85175579705&origin=resultslist&sort=plf-f&src=s&sid=37590ee2bbeec52f39fc5ddb825398f5&sot=b&sdt=b&s=TITLE-ABS-KEY%28pose+estimation%29&sl=29&sessionSearchId=37590ee2bbeec52f39fc5ddb825398f5

[5] Xin Feng, Youni Jiang, Xuejiao Yang, Ming Du, and Xin Li. Computer vision algorithms and hardware implementations: A survey. https://www.sciencedirect.com/science/article/pii/S0167926019301762

[6] Asharul Islam Khan, Salim Al-Habsi, Machine Learning in Computer Vision. https://www.sciencedirect.com/science/article/pii/S1877050920308218

[7] Barney, Nick. Face Detection. (2018) Retrieved from: https://www.techtarget.com/searchenterpriseai/definition/face-detection

[8] Jaiswal, A. (July 25) Face Recognition System. Retrieved from: https://www.analyticsvidhya.com/blog/2022/04/face-recognition-system-using-python/

[9] Singh, M. (2022) A Comprehensive Guide on Human Pose Estimation. Retrieved from: https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/

[10] Odemakinde, E. Pose Estimation Ultimate Overview Retrieved from:

https://viso.ai/deep-learning/pose-estimation-ultimate-overview/

Appendices

1. Documento de requerimientos de software. https://docs.google.com/document/d/1oy6DMTkxfGF-lkOhHSvRgkvQyKZxw91Gbp4GGztZAkQ/edit?usp=sharing
2. Memorandum of Understanding (MOU). https://docs.google.com/document/d/1NB753Xv5pJcMWwVuPgQOgsE-QoMzp6crApv00euU6kM/edit
3. Recursos y metodología. https://docs.google.com/document/d/1o0ZFA7jzRX8eExQ16Q37yHWVRXk0O_yo65zf21vK5R0/edit?usp=sharing
4. Evidencias de comunicación. https://docs.google.com/document/d/1X4DocWQJ_EFGQN72RJdWkf8j-ll4KNYzT1MrofMsIfU/edit?usp=sharing
5. Metrics calculation for the models used. https://colab.research.google.com/drive/1vMVlOhEuuVVqe0N1iCZobFAmGlF0yvJP?usp=sharing