Inteligencia artificial avanzada para la ciencia de datos II (Gpo 501)

Innovación en la Gestión de Asistencias y Participaciones en Aulas Multitudinarias: Un Enfoque Pionero en Detección de Rostros y Reconocimiento de Postura con YOLOv8 y Deep Learning

Autores:

Juan Pablo Castañeda Serrano | A01752030
Aldo Daniel Villaseñor Fierro | A01637907
Francisco Castorena Salazar | A00827756
José Alfredo García Rodriguez | A00830952

Profesores de Reto:

Ivan Mauricio Amaya Contreras
Edgar Covantes Osuna
Hugo Terashima Marìn

Noviembre 2023

Índice

1. Abstract

Este proyecto de reconocimiento facial y seguimiento de participación en entornos educativos tiene como objetivo mejorar la gestión de la asistencia y participación en clases mediante el análisis de video. Utiliza tecnologías avanzadas como YOLOv8 para la detección de personas y estimación de pose, y Face Recognition para el reconocimiento facial. La integración de estas herramientas permite la identificación precisa de estudiantes, asociando sus participaciones individuales durante las clases. Los datos recopilados se almacenan en una base de datos MongoDB y se visualizan a través de una interfaz web, proporcionando estadísticas detalladas para administradores, profesores y estudiantes. La metodología incluye la conexión a la base de datos, carga de caras conocidas, reconocimiento de caras en el video, visualización en tiempo real y eliminación del video original. El proyecto busca ofrecer una solución automatizada y objetiva para mejorar la evaluación del compromiso académico y fomentar la participación activa de los estudiantes.

2. Introducción

El propósito principal de este proyecto es optimizar la gestión de la asistencia y participación en clases mediante el empleo de procesamiento y análisis de video. El objetivo final es desarrollar un sistema que no solo realice las funciones descritas anteriormente, sino que también sea novedoso e innovador, mejorando el rendimiento de los sistemas convencionales de registro de asistencias y participaciones a través de la automatización mediante herramientas tecnológicas. Se sostiene la creencia de que, con las herramientas tecnológicas disponibles en la actualidad, es factible implementar un sistema automatizado que permita mejorar el análisis del rendimiento individual en el aula. Este enfoque personalizado se basa en estadísticas concretas, posibilitando la medición directa de diversas acciones relevantes para los participantes en una clase.

A pesar de que se han propuesto varios sistemas y modelos para automatizar la toma de asistencias y evaluar el rendimiento de los estudiantes en el aula, la literatura presenta una amplia gama de algoritmos y programas para implementar estos sistemas. Un ejemplo de esto es el trabajo de Huayi Zhou et al. [1], que utiliza el algoritmo Region-based Fully Convolutional Network (R-FCN) para la detección de levantamiento de manos y Part Affinity Fields (PAF) para la detección de poses. Por otro lado, para el reconocimiento facial de los alumnos, existen diversas soluciones en la literatura, como la propuesta por Trabelsi et al. [2], que emplea el método de la cascada de Haar.

Sin embargo, es importante destacar que este artículo se enfoca principalmente en el reconocimiento de comportamientos y estados emocionales de los estudiantes, y por lo tanto, no se proporcionan métricas sobre el rendimiento del algoritmo en el reconocimiento facial. Aunque es comúnmente conocido que los algoritmos basados en redes neuronales convolucionales tienden a tener un mejor rendimiento debido a su capacidad para reconocer patrones de datos más complejos y su resistencia a cambios en el entorno, como variaciones en la iluminación o la profundidad, no se proporcionan métricas específicas en este contexto.

Vale la pena señalar que, en general, los algoritmos de reconocimiento facial basados en redes neuronales convolucionales suelen requerir una cantidad significativamente menor de imágenes de entrenamiento en comparación con otros modelos, como LBPH (Local Binary Pattern Histogram). En la tesis presentada por Delbiaggio N. [3] que compara el desempeño de distintos algoritmos de reconocimiento facial, se abordó una prueba de reconocimiento facial para 15 individuos, cada uno representado por 10 imágenes de sus respectivas caras, se encontró que el algoritmo de OpenFace para el reconocimiento facial (basado en redes neuronales convolucionales) logró una tasa de clasificación correcta del 80 %, mientras que LBPH solo logró el 10 % de casos correctamente clasificados.

Dada la diversidad de métodos y soluciones planteadas, es plausible anticipar diferencias en el desempeño de estos sistemas cuando se evalúan en entornos reales y dinámicos. Desafíos como la variabilidad de los datos debido a factores como la baja resolución de imagen en áreas clave, cambios en la iluminación del entorno, presencia de objetos que puedan obstruir la detección de personas, y diversas formas de levantar la mano

son algunos de los retos que se deben tener en cuenta al medir el rendimiento de estos sistemas. Por ende, la evaluación técnica de la viabilidad de los modelos propuestos en situaciones reales requerirá un análisis exhaustivo.

En secciones posteriores de este documento, específicamente en la sección del **Método Propuesto**, se abordará una evaluación del rendimiento del modelo frente a métricas definidas, considerando estos desafíos y proporcionando un marco técnico para determinar la efectividad de los modelos en escenarios prácticos.

A pesar del marco temporal y las restricciones propias de un proyecto universitario de 10 semanas, se aspira a ofrecer una contribución que no solo ofrezca viabilidad técnica, sino que también arroje claridad sobre la intersección y convivencia de diferentes tecnologías de emergencia y vanguardia, como Computer Vision, Cloud Computing e IA. Se busca proporcionar tanto a nosotros los autores como a los lectores, una comprensión más amplia y clara de las herramientas utilizadas, así como de las limitaciones y posibilidades que ofrece esta convergencia tecnológica en el contexto de la educación.

En las secciones posteriores del documento, se buscará primero dar al lector un contexto teórico y general acerca de herramientas fundamentales en Computer Vision dando énfasis en aquellas que serán utilizadas en este proyecto, como reconocimiento facial, detección de poses y YOLO, exponer que herramientas se utilizarán para nuestro sistema y justificar su uso. Se abordará también la implementación de una terminal interactiva diseñada con Flask para facilitar la interacción del usuario. Este conjunto de herramientas establece las bases esenciales para el desarrollo del sistema. Seguido de esto, se profundizará en el funcionamiento del método propuesto, describiendo el diseño del sistema y la integración de las tecnologías mencionadas anteriormente. A continuación, la metodología se sumergirá en la prueba del sistema ante situaciones comunes y experimentales, permitiendo definir resultados y evaluar desempeños. Se culminará con conclusiones sobre la viabilidad del sistema creado, abordando logros, limitaciones y oportunidades de mejora.

3. Fundamentos teóricos

El reconocimiento facial es uno de los problemas recurrentes en el campo de reconocimiento de patrones de visión computacional [**?**] En 1994 Vaillant et al [**?**]plicó una algoritmo de deep learning para el problema de reconocer rostros, propusieron un modelo que era capaz de determinar si había o no un rostro presente en una imagen al haber entrenado un modelo de deep learning.

Deep Convolutional Neural Networks (DCNNs) son ahora el estado del arte en cuanto a reconocimiento facial en vision computacional [**?**]demás de que han alcanzado un desempeño a nivel de un humano e tareas de identificación facial en situaciones realistas [**?**]

3.1. Computer Vision

En si, todo el proyecto se encuentra si no completamente al menos una parte de el es abarcada por un área de la computación que se llama 'visión' y es una constante a lo largo de todo el desarrollo del proyecto, y específicamente la mayor parte del proyecto stá soportada por redes neuronales convolucionales, que son un pilar fundamental a la hora de extraer características de imágenes y a continuación vamos a explicar las partes fundamentales de una Convolutional Neural Network (CNN).

3.1.1. Convolucion

La convolución es una operación matemática que se utiliza para combinar dos matrices. En el contexto de las redes neuronales convolucionales (CNN), la convolución se utiliza para extraer características de las imágenes.

La operación de convolución se puede expresar matemáticamente de la siguiente manera:

$$F(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w_{ij} x_{x+i} y_{y+j}$$

donde:

- $F(x,y)$ es el valor en la posición (x, y) del feature map
- w_{ij} es el peso en la posición (i, j) del kernel
- x_{x+i} es el valor en la posición $(x+i)$ de la matriz de entrada
- y_{y+j} es el valor en la posición (y+j) de la matriz de entrada
- k es el tamaño del kernel

Hablando de imágenes, una característica es un patrón que puede ser útil para identificar un objeto o una clase de objetos. Por ejemplo, las esquinas, los bordes y los colores son características que pueden ser utilizadas para identificar objetos en una imagen [4].

La convolución se realiza deslizando una matriz de pesos, llamada kernel, sobre la matriz de entrada. El kernel se multiplica por cada elemento de la matriz de entrada, y el resultado se suma. El resultado de la convolución se denomina feature map.

Las características extraídas por la convolución se utilizan para entrenar la CNN. La CNN aprende a asignar pesos a las características que son más relevantes para la tarea que se está realizando [4]. Por ejemplo, una CNN que se utiliza para reconocer objetos en imágenes puede aprender a asignar pesos más altos a las características que son comunes a los objetos que se desea reconocer.

Ejemplo [4]:

Consideremos la siguiente imagen:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

Si utilizamos un kernel de 3x3 con los siguientes pesos:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

El resultado de la convolución será el siguiente:

$$\begin{bmatrix} 13 & 22 & 31 \\ 46 & 65 & 84 \\ 79 & 108 & 139 \end{bmatrix}$$

Este feature map representa la suma de los pesos del kernel con los valores de la matriz de entrada en cada posición. Por ejemplo, el valor 13 en la primera fila y la primera columna del feature map es la suma de los siguientes valores:

$$1 * 0 + 2 * 1 + 3 * 2 = 13$$

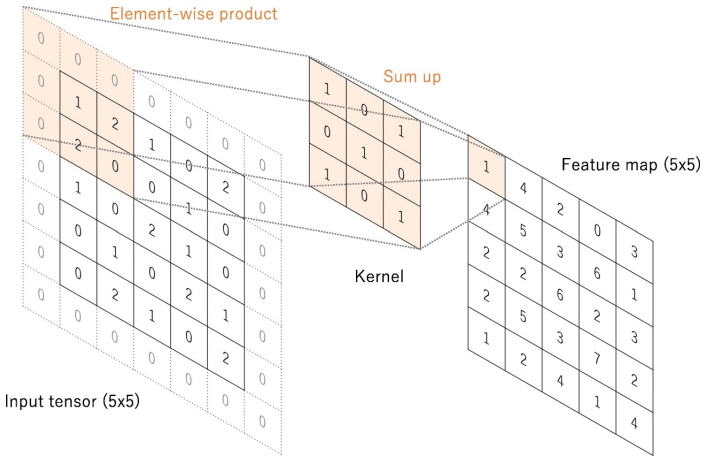

Figura 1: Ejemplo de convolución

3.1.2. Pooling

Las capas de pooling son componentes fundamentales en las redes neuronales convolucionales (CNN). Su función principal es reducir la dimensionalidad espacial de cada mapa de características, lo que ayuda a controlar el número de parámetros y el cómputo en la red, al tiempo que mantiene la información relevante además de dotar de invariabilidad al modelo [5].

Existen dos tipos comunes de capas de pooling: max pooling y average pooling. El max pooling [6] opera dividiendo el mapa de características en regiones y conservando únicamente el valor máximo de cada región, mientras que el average pooling calcula el promedio de los valores en cada región.

Estas capas se utilizan para disminuir el tamaño espacial (ancho y alto) de las representaciones de las características, manteniendo las características más importantes. Reducir el tamaño también ayuda a controlar el sobre ajuste (overfitting) al tiempo que disminuye la carga computacional.

3.1.3. Funciones de activación

Los resultados de una operación lineal, como la convolución, pasan luego por una función de activación no lineal. Aunque en el pasado se usaron funciones no lineales suaves, como la sigmoide o la tangente hiperbólica (tanh), debido a que representan matemáticamente el comportamiento de una neurona biológica, la función de activación no lineal más comúnmente utilizada en la actualidad es la unidad lineal rectificada (ReLU), que simplemente calcula la función: $f(x) = max(0, x)$ [7].

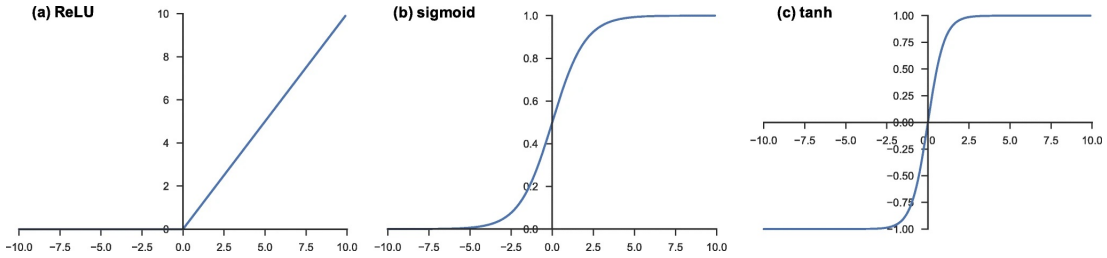

Figura 2: Ejemplo de funciones de activación

3.1.4. Capas totalmente conectadas

Los feature maps resultantes de la capa final de convolución o agrupación se suelen aplanar, es decir, transformadas en un arreglo unidimensional (1D) de números (o vector), y conectadas a una o más capas completamente conectadas, también conocidas como capas densas, en las que cada entrada está conectada a cada salida mediante un peso [7]. Una vez creadas las características extraídas por las capas de convolución y reducidas por las capas de agrupación, son mapeadas por un subconjunto de capas completamente conectadas hacia las salidas finales de la red, como las probabilidades para cada clase en tareas de clasificación. La capa completamente conectada final suele tener el mismo número de nodos de salida que el número de clases. Cada capa completamente conectada está seguida por una función no lineal, como ReLU, como se describió anteriormente y se puede ver en la Figura 2.

3.2. Face Recognition

El reconocimiento facial es una tecnología que se ha utilizado durante muchos años en una variedad de aplicaciones, como seguridad, control de acceso y marketing. En los últimos años, el reconocimiento facial ha experimentado un rápido desarrollo, gracias al aumento de la potencia de procesamiento y la disponibilidad de datos de entrenamiento.

Los primeros sistemas de reconocimiento facial se basaban en técnicas de reconocimiento basadas en características, que identifican a las personas basándose en características faciales específicas, como la forma de la nariz, la boca o los ojos. Sin embargo, estas técnicas eran limitadas en su precisión y no podían funcionar bien en condiciones de iluminación variable o en presencia de cambios en la apariencia facial, como el crecimiento de la barba o el uso de gafas.

En los últimos años, se han desarrollado técnicas de reconocimiento facial basadas en aprendizaje profundo, que han demostrado ser mucho más precisas que las técnicas basadas en características. Estas técnicas aprenden a identificar a las personas a partir de una gran cantidad de datos de entrenamiento, lo que les permite funcionar bien en una variedad de condiciones.

3.2.1. Face Detection con HOG

La detección de rostros es una tarea importante en el procesamiento de imágenes y visión por computadora. Tiene una amplia gama de aplicaciones, como reconocimiento facial, vigilancia y análisis de emociones.

Los primeros métodos de detección de rostros se basaban en características locales, como los puntos característicos de Harris o los descriptores de esquinas. Estos métodos eran relativamente lentos y no eran muy robustos a la variación de iluminación o pose.

En 2001, Viola y Jones [8] presentaron un nuevo método de detección de rostros basado en descriptores HOG (Histogram of Oriented Gradients). Los descriptores HOG son una forma de representar la textura de una imagen. Son robustos a la variación de iluminación y pose, y pueden calcularse rápidamente [8].

El detector de rostros HOG funciona de la siguiente manera:

1. La imagen se divide en una cuadrícula de celdas.
2. Para cada celda, se calcula un descriptor HOG.
3. Los descriptores HOG de cada celda se combinan para formar un vector de características.
4. El vector de características se clasifica como 'rostro' o 'no rostro' utilizando un clasificador.

Los descriptores HOG se calculan contando la cantidad de gradientes de dirección en cada celda. Los

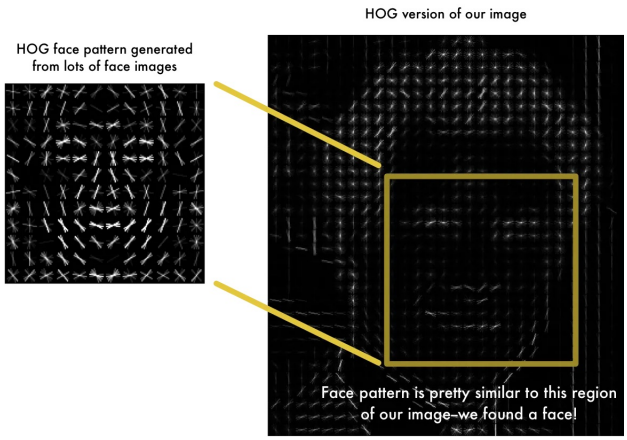

Figura 3: Ejemplo de un diagrama generado con HOG [9]

gradientes de dirección son las direcciones en las que cambian los valores de intensidad de los píxeles en una imagen [8].

Los descriptores HOG son robustos a la variación de iluminación porque son independientes de la magnitud del gradiente. También son robustos a la variación de pose porque son invariantes a la rotación y escala.

3.3. Pose Estimation

La estimación de pose es la tarea de determinar la posición y orientación de las partes del cuerpo de una persona en una imagen o video. Esta tarea es importante para una variedad de aplicaciones, como reconocimiento de gestos, análisis de movimiento y seguimiento de personas.

Los primeros sistemas de estimación de pose se basaban en técnicas de reconocimiento de características, que identifican las partes del cuerpo de una persona basándose en características específicas, como la forma de los hombros, las caderas o las rodillas. Sin embargo, estas técnicas eran limitadas en su precisión y no podían funcionar bien en condiciones de baja iluminación o en presencia de ruido.

La estimación de poses se refiere a la tarea de identificar la ubicación precisa de puntos específicos en una imagen, conocidos como keypoints [10]. Estos puntos clave suelen representar partes distintivas u características relevantes de un objeto, como articulaciones o puntos de referencia significativos. Estos keypoints se expresan habitualmente mediante coordenadas 2D [x, y] o 3D [x, y, visible], siendo fundamentales para comprender la posición relativa o la estructura de un objeto dentro de una imagen [10].

3.4. YoloV8 (You Only Look Once)

La detección de rostros es una tarea importante en el campo de la visión artificial, con aplicaciones en una amplia gama de áreas, como seguridad, reconocimiento facial, análisis de video y marketing. Tradicionalmente, la detección de rostros se ha abordado mediante el uso de técnicas de aprendizaje automático supervisado, que requieren un conjunto de datos de entrenamiento con etiquetas de rostros. Sin embargo, estas técnicas pueden ser lentas y requieren grandes cantidades de datos de entrenamiento.

En 2015, Joseph Redmon y colaboradores publicaron el primer modelo YOLO (You Only Look Once), que representó un avance significativo en el campo de la detección de objetos [12]. YOLO es un modelo de aprendizaje profundo que puede detectar objetos en imágenes y videos en tiempo real. El modelo divide la imagen en una cuadrícula y predice, para cada celda de la cuadrícula, la probabilidad de que contenga un

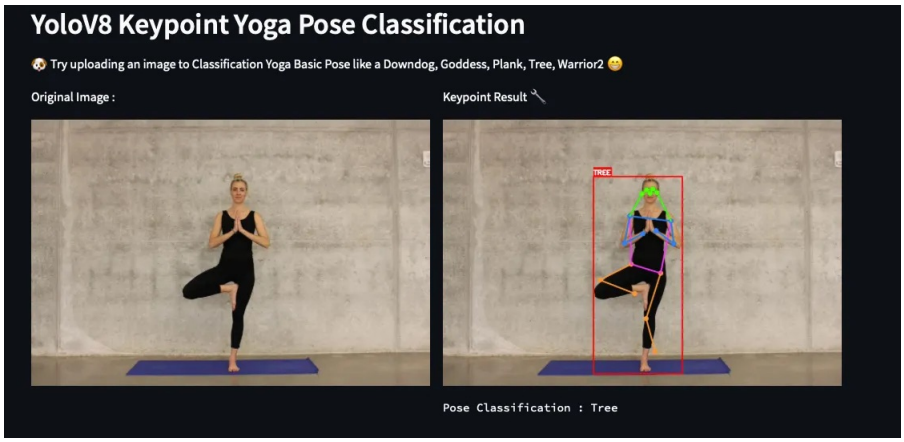

Figura 4: Ejemplo de YOLO-pose en posturas de yoga [11]

objeto, así como las coordenadas de la caja delimitadora del objeto y su clase [13].

YOLO se caracteriza por su velocidad y precisión. En comparación con otros métodos de detección de objetos, YOLO es significativamente más rápido [12], lo que lo hace adecuado para su uso en aplicaciones en tiempo real.

4. Método Propuesto

El sistema propuesto para el registro de asistencias y participación de alumnos en el aula, se compone principalmente de tres partes, el registro de asistencia por medio de algoritmos de reconocimiento facial, registro de participaciones por medio de algoritmos de detecciones de pose y reconocimiento facial y por último el almacenamiento de datos e interfaz para acceder a estos, para este último se integran distintos servicios y aplicaciones que permitan al usuario no solo guardar los datos de las clases, tales como los videos y registros de alumnos, sino también una variedad de features, como opciones para mostrar estadísticas de una clase o alumno, registrar alumnos nuevos, grabar clases y procesar el video por medio del backend que realiza las dos primeras partes mencionadas de este sistema.

4.1. Proceso de Reconocimiento Facial

El algoritmo de reconocimiento facial implementado en backend utiliza la biblioteca face_recognition en Python para detectar y reconocer rostros en los videos. El algoritmo sigue un proceso de tres pasos:

1. Detección de rostros: el algoritmo primero emplea el algoritmo HOG (Histograms of Oriented Gradients) para ubicar rostros dentro de las imágenes que se reciben para ser procesadas.
2. Codificación de rostros: una vez que se detectan rostros, el algoritmo extrae codificaciones para cada rostro. Estas codificaciones representan los rasgos faciales únicos de cada individuo.
3. Reconocimiento facial: las codificaciones extraídas se comparan con una base de datos de codificaciones faciales conocidas. Si se encuentra una coincidencia, el nombre de la persona correspondiente se asocia con la cara detectada.

El algoritmo procesa frames de video, actualiza las personas reconocidas y registra su asistencia en una base de datos MongoDB.

4.2. Algoritmo de registro de Participaciones

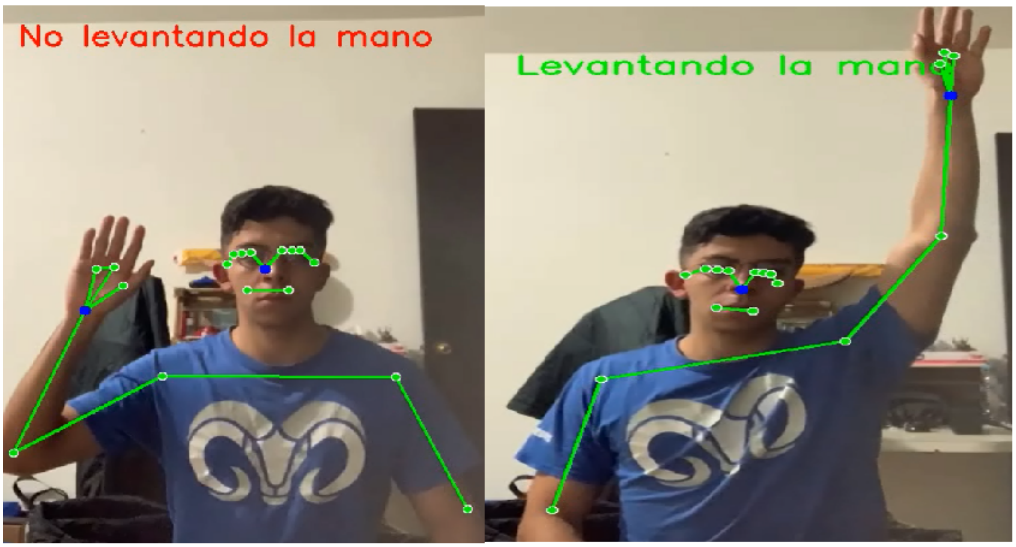

Figura 5: Ejemplo de mano no levantada (izquierda) y mano levantada (derecha).

Primeramente, se establece una condición previamente definida para determinar si una persona está levantando la mano o no. Para lograr esto, se identifican los keypoints de la persona usando YOLO, centrándose especialmente en las coordenadas de los puntos correspondientes a las muñecas y la nariz. Después se define que se está levantando la mano bajo la siguiente lógica: si la coordenada en el eje Y de cualquiera de las muñecas de la persona está por encima de la coordenada en el eje Y de la nariz de la persona, entonces se considera que la persona está levantando la mano.

Como se muestra en la imagen de la izquierda de la figura 5, el punto de la muñeca (marcado en azul) no está en una posición más alta que el punto de la nariz (también en azul), lo que indica que no se considera una posición de manos levantadas. En contraste, en la imagen de la derecha, el punto de la muñeca está por encima del punto de la nariz, lo que sugiere que la persona está levantando la mano. Es importante destacar que para que se registre una levantada de mano como participación, se establece un threshold de frames consecutivos en los cuales la persona debe mantener la mano alzada. En este contexto, se ha definido que 20 frames consecutivos son suficientes para considerarse como una participación. Con esta condición surge un problema, y es que si un alumno permanece con las manos levantadas por dos veces o más la cantidad de frames que se indica como threshold se tomarán participaciones extra. Es por esto que es necesario realizar un seguimiento de los alumnos que tenían la mano levantada previo a que el número de frames supere el threshold, si el alumno ya tenía como stage la mano alzada significa que la participación ya fue tomada por lo cual no es necesario volver a registrarla. Para lograr estos solo se tiene que rastrear quienes son los alumnos que dejan de tener la mano levantada de un frame a otro, de esta forma no es necesario realizar el reconocimiento facial para todos los alumnos en cada frame para verificar que la muñeca se encuentra debajo de la nariz.

Una vez definidas las reglas de lógica y umbrales para nuestro algoritmo, lo primero que se hace es realizar tracking sobre los cuerpos de las personas para poder identificar en que partes de cada frame se encuentran estas, esto se realiza utilizando YoloV8, una vez definidas las áreas que encierran a cada persona, se procede a obtener los keypoints de cada una de las personas utilizando también Yolo.

Frame tras frame se evaluará para cada persona si una de sus muñecas está por encima de su nariz, si se cumpliera esta lógica, se llamará al proceso de reconocimiento facial definido y explicado en la subsección anterior para identificar de los alumnos quien es aquel que está levantando la mano.

Seguido de esto, simplemente se tiene que cumplir un umbral de 20 frames para que se registre la participación en el sistema y sea enviado a la base de datos como una participación por parte de un alumno específico.

En la figura 6 se puede observar un diagrama de flujo que resume el proceso mediante el cual se registran las participaciones.

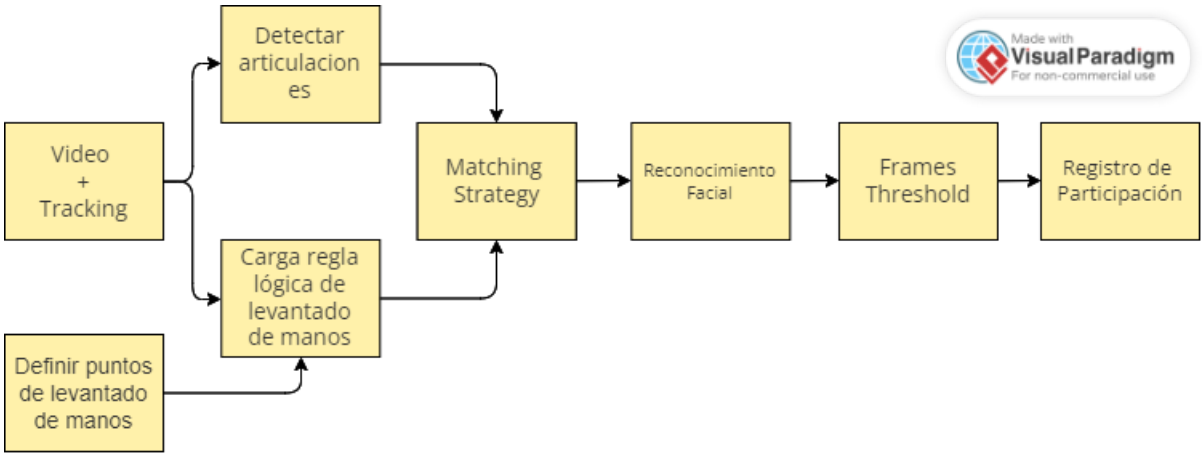

Figura 6: Diagrama de flujo de pasos a seguir por el algoritmo de registro de participaciones implementado.

4.3. Interfaz de Usuario

La interfaz de usuario se realizó utilizando react [14], librería open source en JavaScript utilizada para crear interfaces de usuario interactivas de forma sencilla. Por otro lado se utilizó AWS Amplify para desplegar el frontend de nuestro sistema.

4.4. Backend

El backend se realizó utilizando Flask, Flask proporciona una base sólida para construir aplicaciones web en Python, permitiendo elegir las herramientas y bibliotecas que mejor se adapten a necesidades específicas. Su enfoque minimalista permite un marco flexible y sin complicaciones para el desarrollo web con Python. [15]

El backend se desplegó utilizando Ngrok, Ngrok es una herramienta que se utiliza para crear túneles seguros desde una red local hasta Internet. Su propósito principal es proporcionar una URL pública accesible desde cualquier lugar, incluso fuera de la red local [16], para servicios o aplicaciones que se ejecutan localmente en tu máquina. Cabe mencionar que en caso de querer escalar el sistema de asistencia y participación, se deberá de implementar algún tipo de instancia en la nube como lo podría ser un EC2 de AWS, sin embargo, debido a que a momento este sistema no ha sido implementado de forma real y solo con fines de prueba, Ngrok funciona bien para nuestro prototipado.

Por otra parte, se utilizó AWS Cognito para la creación de usuarios, AWS Cognito se encarga de crear y manejar credenciales de acceso de forma segura, Cognito implementa funciones básicas para el manejo de credencial como el acceso denegado ante credenciales incorrectas o inexistentes, véase figura 7.

Se crearon 3 tipos de usuarios, alumnos, profesores y administradores. Para cada tipo de usuario se implementaron ciertos permisos y restricciones dentro de la interfaz, el administrador tiene permiso para acceder a todas las secciones de la interfaz, por otro lado como se muestra en la figura 8 los estudiantes estaban limitados a únicamente ciertas secciones de la interfaz, en específico a todos menos a la sección

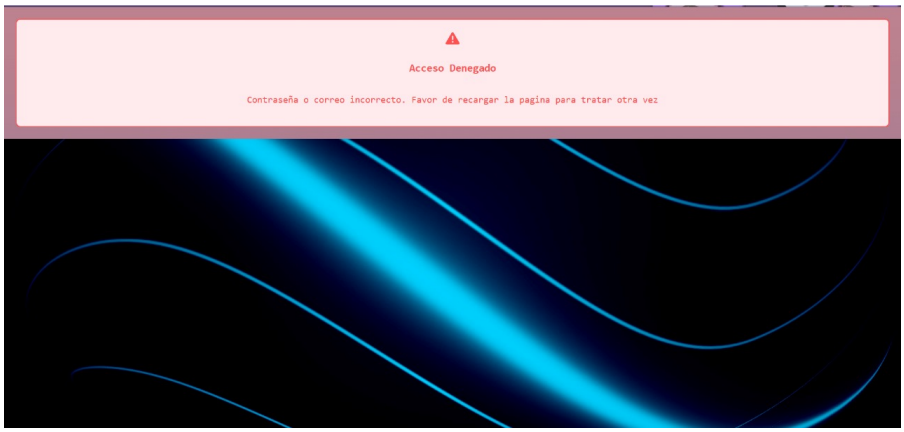

Figura 7: Aviso de Login Fallido debido a credenciales incorrectas

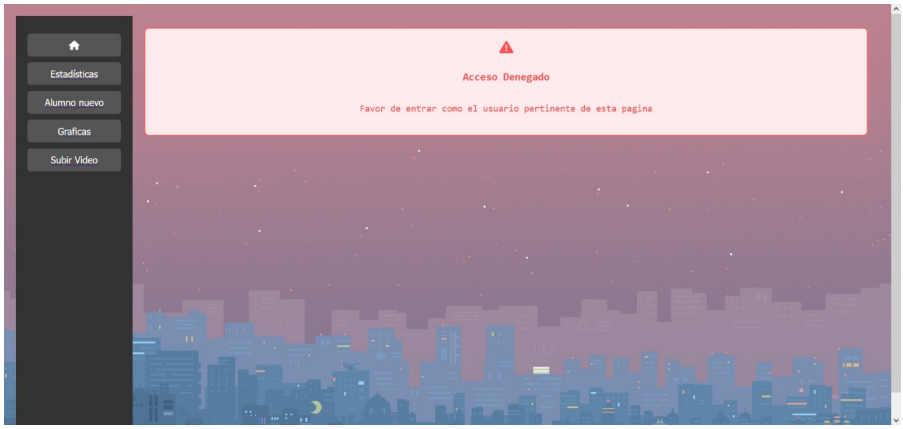

Figura 8: Acceso denegado a usuario del tipo estudiante al tratar de acceder a la sección de Grabar Clase.

Subir Video. De manera similar los profesores pueden acceder a todas las secciones de la interfaz menos a **Alumno Nuevo**.

4.5. Base de Datos

Se utilizó MongoDB para almacenar datos, es una base de datos NoSQL que utiliza un modelo de documentos flexibles en formato BSON.[17]

Ventajas de MongoDB:

- Esquema flexible.
- Escalabilidad horizontal.
- Consultas potentes.

Se desplegó la base en una ip pública con contraseña utilizando MongoDB Atlas. MongoDB Atlas es el servicio en la nube de MongoDB que facilita el despliegue, gestión y escalabilidad de clústeres de MongoDB. Ofrece una interfaz gráfica, escalabilidad automática y funciones avanzadas de seguridad. [18]

4.5.1. Enfoque de Big Data al utilizar MongoDB

En nuestro enfoque hacia Big Data, justificamos la clasificación de nuestro proyecto bajo esta categoría. Al almacenar diversos tipos de datos, como videos, imágenes y bases de datos estructuradas, en forma de cadenas de bytes, cumplimos con los principios fundamentales de Big Data. Esta inclusión de datos heterogéneos y de gran volumen se alinea con la esencia del enfoque Big Data, permitiéndonos gestionar y analizar eficientemente conjuntos de información grandes y diversificados.

4.6. Justificación sobre los métodos seleccionados

Las distintas herramientas utilizadas para implementar el sistema de registro de asistencias y participaciones descrito, fueron seleccionadas en gran parte debido al flujo de trabajo seguido, al definir los objetivos y funciones que el equipo quería que tuviera el sistema a implementar, se realizó una investigación sobre el estado del arte y también se recibieron recomendaciones por parte de profesores y otros compañeros de la clase.

Se propusieron y diseñaron varios métodos para realizar tareas específicas dentro del sistema antes de llegar a la versión final que hoy se explica en el documento, conforme se avanzó en el desarrollo del sistema se encontraron nuevos desafíos o perspectivas que hicieron al equipo reevaluar el sistema que se tenía en ese momento conforme a los objetivos definidos al principio del proyecto, se evaluó el desempeño de distintas librerías, frameworks, scripts, etc. y en base a estos resultados se dio paso a modificar el sistema varias veces, a continuación se muestra una lista de las herramientas y métodos finales que se implementaron junto a la justificación de por qué se decidió utilizar esas soluciones en concreto.

- Uso de Yolo para hacer tracking de personas: Se decidió de forma relativamente rápida que Yolov8 sería el encargado de hacer la detección de personas, esto después de investigar y llegar a la conclusión de que Yolov8 es un modelo del estado de arte para realizar detección de objetos, tracking y pose estimation [19]. Aún y que existen otros modelos igualmente muy buenos como SSD (Single Shot Detector) para este tipo de tareas, se decidió usar Yolo por la gran variedad de artículos e información disponible en internet, así como la comunidad que utiliza y conoce esta herramienta.
- Uso de Yolo para identificar landmarks de las personas: primeramente se optó por utilizar MediaPipe, un framework de Computer Vision realizado por Google AI [20], se implementó un modelo funcional, sin embargo lento debido a que MediaPipe está originalmente diseñado para trabajar con una sola persona en video, se decidió explorar la solución de Yolo para obtener landmarks [21], la cual resultó ser más rápida.
- Uso de algoritmo HOG para realizar face detection: En los primeros modelos se realizaba la detección de rostros por medio del algoritmo de la Cascada de Haar de la librería de opencv-python, aún y que este algoritmo es rápido de computar comparado con otros algoritmos, C Rahmad et al. [22] concluye que el algoritmo de HOG es más preciso que Haar Cascade para realizar detección de rostros. Cabe mencionar que en la práctica se identificó que el método de open-cv para la detección de rostros era muy sensible a cambios de ángulos y al alejarse de la cámara, se ha observado más consistencia en la detección de rostros utilizando HOG.
- Implementación de base de datos en MongoDB:

  Se seleccionó MongoDB como la base de datos para nuestro proyecto debido a su capacidad para gestionar eficientemente distintos tipos de datos y realizar operaciones en lote [23], lo que simplifica el almacenamiento de información sin depender de relaciones complejas. La decisión de optar por una base de datos no relacional se fundamenta en la flexibilidad y agilidad que proporciona al tratar con estructuras de datos variadas y realizar múltiples llamadas al backend.

  La elección de MongoDB también se basa en sus sólidas características de seguridad, que incluyen protección de datos en tránsito y en reposo. Esta consideración es esencial para salvaguardar la integridad y confidencialidad de la información procesada por el sistema.

Además, la decisión de utilizar MongoDB se ve respaldada por su integración con Flask a través de Mongoose, facilitando la conexión y manipulación de la base de datos desde el backend.

- Reconocimiento facial utiizando libreria de Python face_recognition: Se optó por la librería face_recognition de Python debido a su sencillez de uso y rendimiento eficiente. La simplicidad en la implementación facilita la integración con nuestro proyecto, permitiendo un desarrollo más ágil y mantenible [24]. Además, la velocidad de face_recognition en la detección y reconocimiento de rostros fue un factor decisivo, brindando resultados precisos de manera rápida.

  Una alternativa comúnmente considerada es OpenCV, sin embargo, se percibió que face_recognition ofrece una interfaz más simple y centrada en el reconocimiento facial, lo que se alinea mejor con los requisitos específicos de nuestro proyecto, cabe mencionar que se hicieron pruebas de reconocimiento facial con OpenCV y se tardaba mucho en procesar los frames, también se realizaba la detección de rostros utilizando la cascada de Haar que era más imprecisa que HOG, por lo que había muchas zonas y ángulos de caras donde no se detectaba una cara.

- Uso de AWS para alojar deploy de la interfaz web: los siguientes factores fueron por los cuales se decidió utilizar los servicios de AWS:

  - Familiaridad: El equipo ya tiene experiencia con los servicios de AWS, facilitando la configuración y gestión.
  - Precio Competitivo: La estructura de precios competitiva y los modelos de pago por uso de AWS se adaptan bien a las necesidades del proyecto.
  - Escalabilidad y Confiabilidad: Soluciones automáticas de escalabilidad y una infraestructura global garantizan alta disponibilidad y confiabilidad. [25]

4.7. Explicación del Sistema Integrado

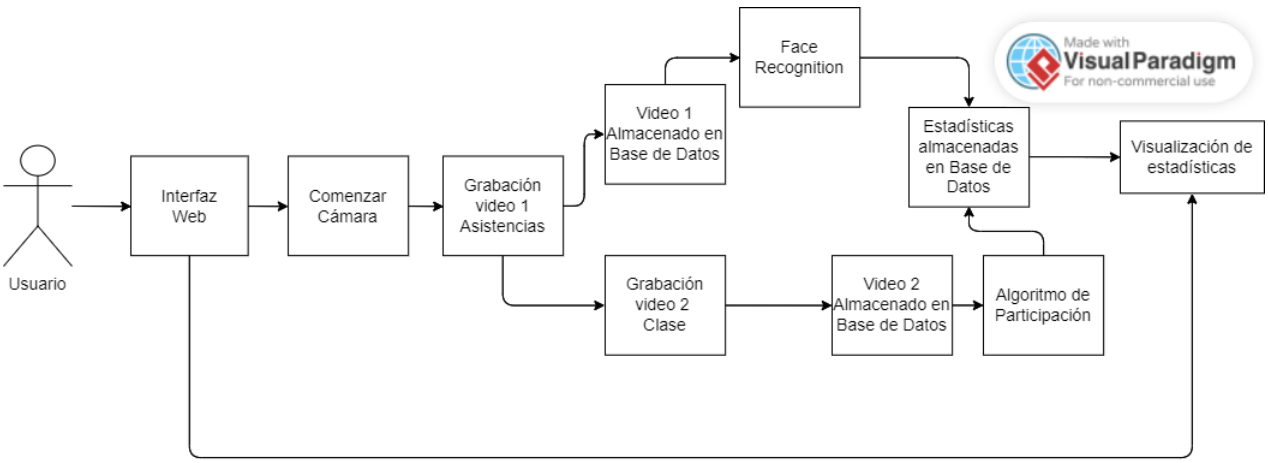

Figura 9: Diagrama de flujo del proceso realizado en el sistema implementado para la toma de asistencias y participaciones en el aula.

Como se muestra en la figura 9, el sistema inicia con la interacción del usuario en una interfaz web. Esta interfaz coordina funciones y servicios para grabar, almacenar, y procesar videos de clases y asistencias. Además, gestiona el registro de nuevos alumnos y la presentación de estadísticas mediante gráficas y valores estadísticos.

Una vez en la interfaz web, el usuario puede activar la cámara para el registro de asistencias y participaciones. Al iniciar la cámara, se debe hacer clic en "Empezar grabación de asistencia para comenzar a grabar el video. Este video se envía al backend para el reconocimiento facial, donde se mapean los puntos

descriptores de las caras con la base de datos de alumnos y profesores. Los participantes deben pasar frente a la cámara durante este proceso.

Una vez que todos han sido registrados, se hace clic en "Empezar grabación de clase, que graba el video de la clase y envía el video de asistencias (video 1) al backend. Este video se almacena en la base de datos y se procesa en paralelo con la grabación de la clase para actualizar las estadísticas. Al finalizar la clase, se hace clic en "Terminar la clase, lo que almacena el video de clase (video 2) en la base de datos.

Posteriormente, el video 2 es procesado por la parte del backend que contiene el algoritmo de registro de participaciones. La información se envía a la base de datos para actualizar las estadísticas respectivas. Este proceso puede tardar algunos minutos según la duración del video.

Una vez que ambos videos se han procesado y los datos se han registrado en la base de datos, el usuario puede acceder a la sección de estadísticas desde la interfaz. Aquí, se presentan gráficas y datos relevantes sobre estudiantes y clases.

5. Metodología

Antes de abordar las pruebas específicas, es crucial establecer un marco general para evaluar el sistema. Este esquema de pruebas debe abarcar tres componentes principales:

1. Pruebas del Sistema de Reconocimiento Facial: El objetivo de estas pruebas será verificar la precisión y confiabilidad del sistema de reconocimiento facial.
2. Pruebas del Sistema de detección de Pose: El objetivo será evaluar la precisión y la efectividad del sistema de detección de posturas, específicamente el levantado de manos.
3. Pruebas de Integración del Sistema: Aquí se busca asegurar que los componentes individuales del sistema se integren de manera coherente.

A continuación en la figura 10 se muestra un overview del esquema de pruebas seguido para verificar la eficacia integral del sistema propuesto. Cabe mencionar que las flechas del esquema describen el orden con que se realizó cada sección de pruebas, primero se evaluó el sistema de detección de rostros debido a que iba a ser utilizado también en el algoritmo de participación, por último ambos procesos se integraron a la interfaz, una vez funcionando de forma satisfactoria.

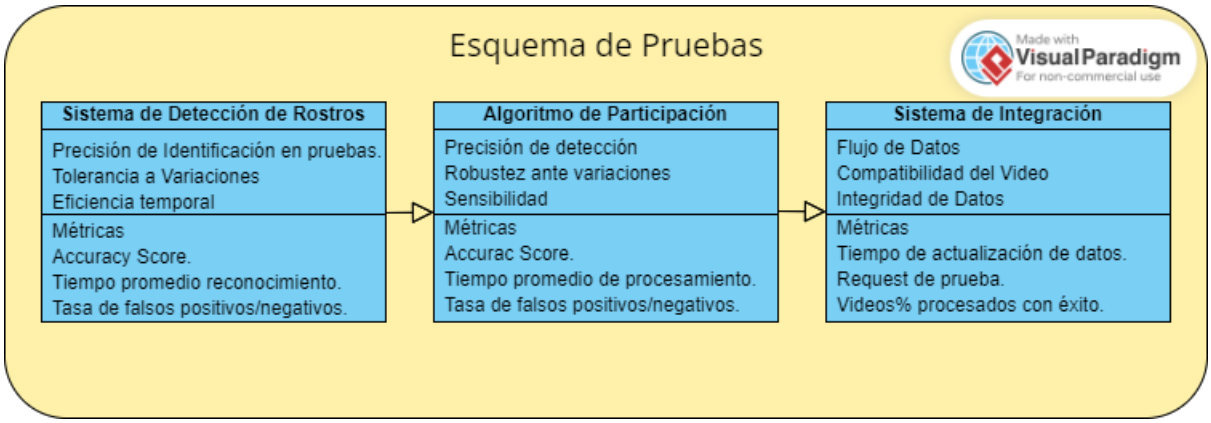

Figura 10: Esquema de Pruebas a seguir para el sistema, en cada bloque se muestra a que sección del sistema pertenece ese bloque, luego los objetivos y finalmente las métricas a utilizar para asegurar estos.

Se mostrarán en las subsecciones siguientes el proceso para realizar y evaluar las pruebas a detalle de cada sección mencionada anteriormente.

5.1. Matriz de Confusión

Para las pruebas de reconocimiento facial y para el algoritmo de participación se tendrán que detectar 2 escenarios fundamentales que nos ayudarán a definir cuando el algoritmo se equivocó, primero se querrá saber cuando el algoritmo detecto caras o poses de forma incorrecta, por ejemplo que haya clasificado la cara de una persona de forma errónea o que un alumno esté levantando la mano pero el algoritmo no lo detecte (Falso Negativo), por otro lado se querrá saber las veces que el algoritmo no tuvo que haber reconocido una cara o que no haya tenido que clasificar una pose como levantado de manos, mas sin embargo lo hizo (Falso Positivo), estas situaciones se pueden clasificar y agrupar de forma muy sencilla por medio de una matriz de confusión, esta es una herramienta utilizada en el campo de clasificación para evaluar el rendimiento de modelos de Machine Learning, Proporciona una visión detallada de la relación entre las predicciones del modelo y las clases reales de un conjunto de datos. La matriz organiza las predicciones en cuatro categorías distintas:

1. Verdaderos Positivos (TP): Casos en los que el modelo predijo correctamente la clase positiva.
2. Falsos Positivos (FP): Casos en los que el modelo predijo incorrectamente la clase positiva cuando la clase real era negativa (error tipo I).
3. Falsos Negativos (FN): Casos en los que el modelo predijo incorrectamente la clase negativa cuando la clase real era positiva (error tipo II).
4. Verdaderos Negativos (TN): Casos en los que el modelo predijo correctamente la clase negativa.

La matriz de confusión es esencial para calcular diversas métricas de evaluación del rendimiento del modelo, como la precisión (accuracy), la sensibilidad (recall), la especificidad y el puntaje F1. También es útil para visualizar patrones de errores y entender cómo el modelo clasifica las diferentes clases en un conjunto de datos.

5.2. Pruebas de Reconocimiento Facial

A lo largo del desarrollo del reto tuvimos varios desafíos que se nos fueron presentando y es por eso que tuvimos que probar entre diferentes soluciones o maneras de acercarnos a la problemática para poder resolver satisfactoriamente el reto, específicamente para el caso de face recognition fue necesario probar varias alternativas de entre las cuales seleccionamos la mejor tomando en cuenta 3 factores principalmente:

1. El tiempo de procesamiento.
2. La proporción de personas detectadas correctamente (Recall).
3. Consistencia del algoritmo en condiciones no favorables (iluminación, ángulo, etc.) (accuracy / exactitud).

Entonces para evaluar nuestro algoritmo de reconocimiento de caras decidimos hacer algo similar a lo que se hace con los algoritmos de clasificación e interpretar los resultados con una matriz de confusión y un reporte de clasificación. Además revisando en la literatura decidimos agregar algunas métricas extras, como:

Tomando las métricas más importantes del reporte de clasificación específicamente para nuestro problema:

Exactitud (Accuracy)

$$\text{Accuracy} = \frac{\text{Número de predicciones correctas}}{\text{Número total de muestras}}$$

Recall (Sensibilidad o Exhaustividad):

$$\text{Recall} = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos} + \text{Falsos negativos}}$$

Puntuación F1:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Y después para comparar los algoritmos lo que se hizo fue tomar 10 imágenes de cada uno de los integrantes (Alfredo, Paco, Juan Pablo y Aldo) para después aplicar el algoritmo de reconocimiento facial y luego comparamos el output con el valor real.

5.3. Pruebas sobre el Algoritmo de Participación

A lo largo del proyecto, se realizaron varios videos de prueba, de varias personas en el aula levantando la mano para detectar sus participaciones.

Se realizó la prueba principal con un video de duración de aproximadamente 3 minutos, el video tenía 5 personas, el video simulaba el levantamiento de las manos de las personas, este video fue realizado en un salón de clases donde el fondo no cambiaba, véase figura 11 y se tenía buena iluminación.

5.3.1. Prueba Video 1 (Fondo Estable)

- Número de Personas: 5
- Duración del Video: 2 minutos, 44 segundos.
- Tiempo de Procesamiento en backend: 4 minutos, 30 segundos.

De observaciones sobre el video se tiene que era una ventaja tener un fondo estable sin interferencias externas, se obtuvo un mejor rendimiento general.

Figura 11: Imagen sobre el Video 1 de Prueba, se observa que el fondo es estable.

5.4. Pruebas para verificar el funcionamiento del Sistema Integrado

En esta sección se busca ver que las distintas partes que conforman nuestro sistema de registro de asistencias y participaciones funcionen bien de forma conjunta. Para esto se verificará que el flujo de datos es correcto, con esto nos referimos a que la interfaz sea capaz de, registrar un nuevo alumno, grabar un video de asistencias, grabar un video de clases, que estos sean almacenados correctamente en base de datos y que después sean procesados en nuestro backend, para que finalmente los registros sobre los alumnos y las clases se actualicen y también se muestren las distintas gráficas e indicadores en la sección de Estadísticas de la UI actualizadas.

A continuación se mostrarán los elementos que serán evaluados para saber si la interfaz funciona de forma correcta.

- Log in correcto a la interfaz web, indicador de los datos de estudiantes siendo cargados a la interfaz.
- Generación correcta de gráficas y estadísticas al ser actualizada la base de datos.
- Proceso de grabar y procesar videos de asistencias llamando al backend desde el servidor en la interfaz de usuario.

6. Resultados

6.1. Sobre Pruebas de Reconocimiento Facial

face_recognition **Matriz de confusión**

	Alfredo	Aldo	Juan	Pablo	Paco	Unknown
Alfredo	9	1	0	0	0	0
Aldo	1	8	1	0	0	0
Juan	0	0	9	1	0	0
Pablo	0	1	0	8	1	0
Paco	0	1	0	0	9	0
Unknown	0	1	2	0	1	6

Tiempo: 2 segundos.

Reporte de clasificación

Clase	**Precisión**	**Recall**	**F1-score**	**Soporte**
Alfredo	0.90	0.90	0.90	10
Aldo	0.67	0.80	0.73	10
Juan	0.75	0.90	0.82	10
Pablo	0.89	0.80	0.84	10
Paco	0.82	0.90	0.86	10
Unknown	1.00	0.60	0.75	10
Exactitud		0.82 (60 muestras)		
Promedio	0.84	0.82	0.82	60

Cuadro 1: Métricas de clasificación

6.1.1. VGG15 - face

Matriz de confusión

	Alfredo	Aldo	Juan	Pablo	Paco	Unknown
Alfredo	10	0	0	0	0	0
Aldo	0	9	1	0	0	0
Juan	0	0	9	1	0	0
Pablo	1	0	0	8	1	0
Paco	0	1	0	0	9	0
Unknown	1	1	2	0	1	5

Tiempo: 5 segundos

Reporte de clasificación

Clase	**Precisión**	**Recall**	**F1-score**	**Soporte**
Alfredo	0.83	1.00	0.91	10
Aldo	0.82	0.90	0.86	10
Juan	0.75	0.90	0.82	10
Pablo	0.89	0.80	0.84	10
Paco	0.82	0.90	0.86	10
Desconocido	1.00	0.50	0.67	10
Exactitud		0.83 (60 muestras)		
Promedio	0.85	0.83	0.83	60

Cuadro 2: Métricas de clasificación

6.1.2. Meta - DeepFace

Matriz de confusión

	Alfredo	Aldo	Juan	Pablo	Paco	Unknown
Alfredo	10	0	0	0	0	0
Aldo	0	9	1	0	0	0
Juan	0	0	9	1	0	0
Pablo	1	0	0	10	0	0
Paco	0	1	0	0	9	0
Unknown	1	1	0	1	0	8

Tiempo 11 segundos.

Reporte de clasificación

6.1.3. Comparación y selección

Podemos ver en los resultados de nuestras evaluaciones que los modelos tuvieron rendimientos bastante similares entre ellos en 4, sin embargo, un aspecto clave en nuestra solución del reto es que planeamos que el procesado sea lo mas ligero y por consecuencia barato que se pueda, esto para que pueda ser implementado sin impactar en el bolsillo del cliente.

Es por eso que tomando en cuenta que el algoritmo que toma mucho menos tiempo que los demás, además de obtener unas métricas buenísimas considerando que no eran las mejores condiciones, cosa contraria a un

Clase	**Precisión**	**Recall**	**F1-score**	**Soporte**
Alfredo	0.83	1.00	0.91	10
Aldo	0.82	0.90	0.86	10
Juan	0.90	0.90	0.90	10
Pablo	0.83	0.91	0.87	11
Paco	1.00	0.90	0.95	10
Desconocido	1.00	0.73	0.84	11
Exactitud		0.89 (62 muestras)		
Promedio	0.90	0.89	0.89	62

Cuadro 3: Reporte de clasificación

Modelo **Promedio F1-score**	**Tiempo (seg)**	**Exactitud**	**Promedio Precisión**	**Promedio Recall**
Modelo 1 (face_recon) 0.82	2	0.82	0.84	0.82
Modelo 2 (VGG15 - face) 0.83	5	0.83	0.85	0.83
Modelo 3 (Meta - DeepFace) 0.89	11	0.89	0.90	0.89

Cuadro 4: Comparación de métricas de rendimiento de modelos

salón de clases en donde el ambiente por lo regular siempre estará en buenas condiciones. Considerando todo lo anterior decidimos decantarnos por el algoritmo de face_recognition que nos brinda de todo lo que necesitamos a un bajo coste computacional.

6.2. Sobre Pruebas de Algoritmo de Participación

A continuación se muestra la matriz de confusión correspondiente al video de prueba.

		Real		
		Positive	Negative	Total
Predicha	Positive	62	0	62
	Negative	16	0	16
	Total	78	0	

En este caso, no se presentan verdaderos negativos debido a que serían demasiados por frame de video, nuestro algoritmo no registra cuando un alumno no está participando, solo cuando este participa, sin embargo podemos darnos una muy buena idea del desempeño tan solo mirando los verdaderos positivos y los Falsos negativos.

Teniendo en cuenta la métrica de accuracy

$$Accuracy = \frac{VP + VN}{VP + VP + FN + TN}$$

Vemos que el accuracy en la prueba fue del 79 %.

Teniendo en cuenta la métrica de precisión

$$precision = \frac{VP}{VP + FP}$$

Se observa que se obtuvo una precisión de 1, por lo que esta métrica no es muy útil en nuestro caso.

6.3. Sobre Pruebas de Integración de Servicios

6.3.1. Log in y Carga de Datos

Para esto se accede al siguiente link https://main.dg4xf9dmaefhh.amplifyapp.com/ el cual hace hosting a la interfaz en una ip pública y se ingresó con las correspondientes credenciales de usuario, una vez realizado esto se registró desde la terminal del servidor de la interfaz estas acciones, véase figura 12

```
 * Debugger is active!
 * Debugger PIN: 937-004-718
127.0.0.1 - - [27/Nov/2023 23:11:00] "OPTIONS /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:00] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:06] "GET /get-students HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:06] "GET /get-students HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:23] "OPTIONS /upload HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:23] "POST /upload HTTP/1.1" 200 -
```

Figura 12: Registro de inicio de sesión y carga de datos de estudiantes a la interfaz.

Nótese también que una vez registrado el login, se realiza un request a la base de datos para cargar los datos de estudiantes y clases a la página web.

6.3.2. Generación de gráficas y Estadísticas

Una vez dentro de la página y cargados los datos de los estudiantes, se accede a la sección de **Estadísticas** para visualizar las gráficas y valores estadísticos respectivos a estudiantes y clases, véase en la figura 13 la carga de las gráficas a la interfaz.

```
<_io.BytesIO object at 0x0000013B25FD9EA0>
127.0.0.1 - - [27/Nov/2023 23:11:33] "GET /generate_heatmap HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:33] "GET /generate_bubble_chart HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:33] "GET /generate_bubble_chart HTTP/1.1" 200 -
<_io.BytesIO object at 0x0000013B25EA1F40>
127.0.0.1 - - [27/Nov/2023 23:11:33] "GET /generate_heatmap HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:45] "OPTIONS /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2023 23:11:45] "POST /login HTTP/1.1" 200 -
```

Figura 13: Registro desde terminal de la generación de gráficas, una vez que se da clic en la sección de **Estadísticas**.

Una vez que se da clic en **Estadísticas** se muestran distintas gráficas respectivas a alumnos y clases, como ejemplo de esto, véase la figura 14 para observar algunas de las gráficas disponibles para los alumnos y la figura 15 para algunas de las gráficas disponibles para las clases.

6.3.3. Procesamiento de Vídeos desde la Interfaz

También se probó desde la interfaz el subir videos de prueba de asistencia y de clase para ver que se llamaba al backend correctamente, una vez grabados los videos, y mandados a ser procesados desde la interfaz, se obtiene el siguiente registro desde terminal, véase figura 16

Nótese que se arrojan los resultados del algoritmo de participación en el diccionario mostrado en la figura 16, debido a que se quizá probar aquí únicamente la conexión exitosa entre interfaz, base de datos y backend, no se tiene registro de participaciones para ningún alumno pero si se observa como fue ejecutado el script de backend en 6.69 segundos.

Figura 14: Ejemplo de gráficas disponibles para alumnos registrados en la interfaz, en este caso se muestran algunas gráficas correspondientes al alumno Paco.

7. Conclusiones

La evaluación del modelo de detección de poses revela un buen rendimiento global con un accuracy del 79 % en el caso de prueba. Este indicador sugiere que el modelo realiza predicciones precisas en la mayoría de los casos. Sin embargo, se observa una tasa significativa de falsos negativos, indicando que algunas participaciones no se detectan de manera óptima, una consideración crítica en un contexto educativo donde cada participación es valiosa.

Como sugerencias de mejora, se propone experimentar con la modificación del umbral de detección para equilibrar la sensibilidad y precisión del modelo, lo que podría reducir la tasa de falsos negativos. Además, explorar modelos alternativos de detección de poses puede proporcionar nuevas perspectivas y mejorar la capacidad del sistema para capturar poses de manera más efectiva.

Una de las partes cruciales del algoritmo es el reconocimiento facial, el cual en ciertas condiciones no logra reconocer al alumno en el frame y le da el nombre de Ünknown". Esto crea oportunidad para mejorar el rendimiento del algoritmo se encuentra en implementar condiciones basadas en la distancia de un alumno detectado como Ünknownçon un alumno que si fue detectado en frames pasado, ya que hay ciertas condiciones en las que el algoritmo falla por errores en el reconocimiento facial, pero que con arreglos en el código podrían solucionarse esos errores aprovechándonos de los supuestos de los que partimos del código y del correcto funcionamiento de los modelos en frames anteriores.

La limitación de recursos computacionales en las computadoras personales de los integrantes del proyecto ha impactado en el tiempo de procesamiento, especialmente al procesar videos de participación. Se sugiere buscar recursos computacionales más sofisticados para implementar algoritmos de detección de poses más robustos y mejorar las velocidades de procesamiento, especialmente si se ejecuta el backend localmente.

Considerando técnicas avanzadas, la implementación de estrategias como aumento de datos y enfoques de transfer learning podría enriquecer la capacidad del modelo para reconocer patrones complejos en la detección de participaciones, mejorando así la generalización del modelo.

En resumen, aunque el modelo actual muestra un rendimiento positivo, las sugerencias de mejora se centran en ajustar parámetros, explorar alternativas y mejorar los recursos computacionales para maximizar la eficacia de la herramienta en un entorno educativo.

La detección de participaciones puede mejorarse mediante ajustes en el umbral, análisis detallado de falsos

Figura 15: Ejemplo de gráficas disponibles para las clases registradas en la interfaz, en este caso se muestran algunas gráficas correspondientes a la clase de IA.

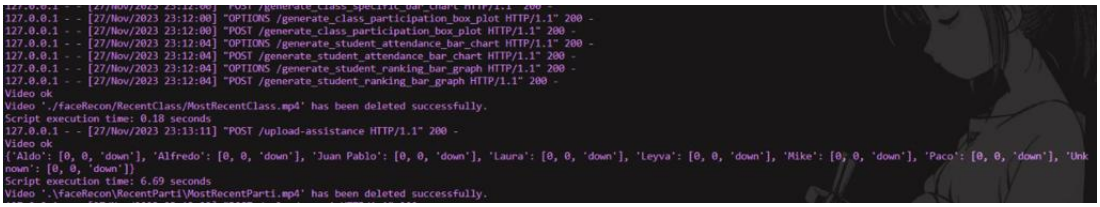

Figura 16: Se observa como se cargan los videos, se ejecuta sus respectivos backends y se eliminan los videos de la base de datos para solo tener finalmente las estadísticas.

negativos y la implementación de estrategias de mejora progresiva. También se considera que la colaboración entre expertos técnicos y educativos es de sumo valor si se quiere mejorar el modelo lo más posible.

Se observó que un denominador común en los falsos negativos en reconocimiento facial eran generados con personas de tonos de piel menos claros, se observó que el algoritmo de reconocimiento de rostros tenía mayores problemas para reconocer rostros de piel menos clara.

De igual manera al realizar la interfaz de interacción humana concluimos que la mejor implementación a largo plazo sería utilizar un servicio de servidores similar a una ec2 de Amazon, donde se procesarían todos los algoritmos realizados en lugar de un servicio tunelless como lo es Ngrok. De cualquier modo, sentimos que esta manera de interfaz con backend en una computadora física funciona para efectos académicos y satisface los requerimientos establecidos por el socio formador.

En resumen, el desarrollo integral de este proyecto ha culminado en la creación de un sistema funcional y cohesivo. La integración exitosa de diversas partes, como el reconocimiento facial, el algoritmo de detección de participaciones, la interfaz de usuario intuitiva y la gestión eficiente de datos en MongoDB, ha resultado en un sistema viable de principio a fin. La ejecución del backend utilizando herramientas como Ngrok ha facilitado la accesibilidad externa a las funcionalidades del sistema. Aunque se reconocen aspectos a mejorar, como la precisión del modelo de detección de poses y la optimización de recursos computacionales, estos desafíos son abordables mediante un enfoque de mejora continua. La flexibilidad y modularidad del sistema permiten implementar ajustes y refinamientos de manera eficiente, respaldando la evolución y la adaptabilidad del proyecto a medida que se avanzan en futuras iteraciones.

Referencias

[1] H. Zhou, F. Jiang, and R. Shen, "Who are raising their hands? hand-raiser seeking based on object detection and pose estimation," in *Proceedings of The 10th Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Zhu and I. Takeuchi, Eds., vol. 95. PMLR, 14–16 Nov 2018, pp. 470–485. [Online]. Available: https://proceedings.mlr.press/v95/zhou18a.html

[2] Z. Trabelsi, F. Alnajjar, M. M. A. Parambil, M. Gochoo, and L. Ali, "Real-time attention monitoring system for classroom: A deep learning approach for studentrsquo;s behavior recognition," *Big Data and Cognitive Computing*, vol. 7, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2504-2289/7/1/48

[3] N. Delbiaggio, "A comparison of facial recognition algorithms," 2017.

[4] "Convolutional Neural Network (CNN)." [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[6] Zhou and Chellappa, "Computation of optical flow using a neural network," in *IEEE 1988 International Conference on Neural Networks*, 1988, pp. 71–78 vol.2.

[7] R. Yamashita, M. Nishio, R. K. Gian, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 6 2018. [Online]. Available: https://doi.org/10.1007/s13244-018-0639-9

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[9] f. given i=A, given=Adam, "Machine learning is fun! part 4: Modern face recognition with deep learning." [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

[10] Ultralytics, "Pose." [Online]. Available: https://docs.ultralytics.com/tasks/pose/

[11] A. Mustofa, "YoloV8 Pose Estimation and Pose Keypoint Classification using Neural Net PyTorch," 8 2023. [Online]. Available: https://alimustoofaa.medium.com/yolov8-pose-estimation-and-pose-keypoint-classification-using-neural-net-pytorch-98469b924525

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[13] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[14] React. (2023-11-27) React. Facebook. [Online]. Available: https://react.dev/

[15] Pallets. (2023-11-27) Flask. [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/

[16] Ngrok. (2023) Ngrok. [Online]. Available: https://ngrok.com/

[17] "Mongodb," https://www.mongodb.com/, accessed: Nov 27, 2023.

[18] "Mongodb atlas," https://www.mongodb.com/atlas/database, accessed: Nov 27, 2023.

[19] Ultralytics, "Ultralytics," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[20] G. AI, "Pose landmark detection guide," 2023. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker/

[21] Ultralytics, "Pose estimation," 2023. [Online]. Available: https://docs.ultralytics.com/tasks/pose/

[22] C. Rahmad, R. A. Asmara, D. R. H. Putra, I. Dharma, H. Darmono, and I. Muhiqqin, "Comparison of viola-jones haar cascade classifier and histogram of oriented gradients (hog) for face detection," *IOP Conference Series: Materials Science and Engineering*, vol. 732, no. 1, p. 012038, jan 2020. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/732/1/012038

[23] MongoDB, "Mongodb documentation," 2023, accessed: Nov 27, 2023.

[24] ageitgey, "face_recognition documentation," https://face-recognition.readthedocs.io/en/latest/readme.html#python-code-examples, 2023, accessed: Nov 27, 2023.

[25] "Amazon web services," https://aws.amazon.com/, accessed: Nov 27, 2023.

8. Anexos

Repositorio de github: https://github.com/a01752030/PortafolioRetoFC

Video demostrativo: https://www.youtube.com/watch?v=QAmuV5ITtFY

Interfaz desarrollada: https://main.dg4xf9dmaefhh.amplifyapp.com/

*En caso de requerir acceso, mandar correo a cualquier integrantes del equipo AXXXXXXXX@tec.mx