Implementación de un sistema de automatización para la gestión de asistencia mediante el uso de reconocimiento facial y de caracteres

Isabella Jiménez Bravo, Melissa Alejandra

York Sánchez y Joshua Rodríguez López

- Universidad de Xalapa - Visión Artificial -

15 de enero de 2025

Keywords— Inteligencia Artificial, Visión computacional, Reconocimiento facial, Reconocimiento de caracteres (OCR), Gestión de Asistencia, Técnicas de aprendizaje automático

Abstract

Artificial intelligence has gained popularity due to its applications in various fields. This work focuses on the implementation of an attendance management system that utilizes facial and character recognition technologies, addressing the challenges of traditional attendance methods that are error-prone and require constant manual intervention.

The proposal includes a methodological approach involving data collection and preparation, image processing, neural network model development, and system performance evaluation. The relevance of computer vision techniques in automating attendance registration is highlighted, as well as their potential to improve the educational and administrative experience in educational and workplace institutions.

Resumen

La inteligencia artificial ha incrementado su popularidad gracias al interés en diversos campos en los que ha sido implementada. Este trabajo se centra en la implementación de un sistema de gestión de asistencia que utiliza tecnologías de reconocimiento

facial y de caracteres, abordando los retos de los métodos tradicionales de registro de asistencia que son propensos a errores y requieren intervención manual constante.

La propuesta incluye una propuesta metodológica que contempla la recolección y preparación de datos, el procesamiento de imágenes, el desarrollo de modelos de redes neuronales y la evaluación del rendimiento del sistema. Se destaca la relevancia de las técnicas de visión computacional en la automatización del registro de asistencia, así como su potencial para mejorar la experiencia educativa y administrativa en instituciones educativas y laborales.

1. Introducción.

La inteligencia artificial (IA) es un campo de estudio que ha estado en auge en los últimos años debido a las aplicaciones de esta en distintos sectores, desde el entretenimiento hasta el sector médico. Su capacidad para transformar procesos, automatizar tareas y resolver problemas complejos ha llamado la atención del público en general, empresas e investigadores.

De acuerdo con Rouhiainen, la inteligencia artificial se define como la capacidad de las máquinas para usar algoritmos, aprender de los datos e utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano [**b2**].

Las aplicaciones técnicas de la IA abarcan una amplia gama de áreas, incluyendo el análisis de datos, detección de patrones, la predicción de eventos o comportamientos, y la clasificación de información, entre otros. Dentro de este vasto campo, han surgido múltiples ramas de investigación especializadas; como la optimización, el cómputo evolutivo, el procesamiento del lenguaje natural y la visión artificial, cada una con aplicaciones y enfoques únicos, aunque estas ramas se pueden combinar.

En particular, la visión artificial es una rama de investigación que permite a las máquinas interpretar y comprender el mundo visual. Esto implica el uso de cámaras para obtener datos de fotos en tiempo real, imágenes o videos, y así, realizar una extracción y análisis de información relevante.

Según García y Caranqui, la visión artificial puede ser definida como los procesos de obtención, caracterización e interpretación de información de imágenes tomadas desde un mundo tridimensional a partir de imágenes bidimensionales [**b3**]. Esta tecnología es aplicada en áreas como la robótica, la medicina y la seguridad.

Dentro de la visión artificial existen distintos subcampos, como el reconocimiento facial y de caracteres. Estos subcampos pueden ser utilizados como soluciones eficaces para la automatización de la gestión de asistencia. Estas tecnologías no solo permiten identificar a los individuos de manera precisa y rápida, sino que también mejoran la experiencia de los estudiantes y optimizan los procesos administrativos en instituciones educativas y laborales.

2. Planteamiento del problema.

La gestión de asistencia en instituciones educativas o laborales a menudo se enfrenta a desafíos, como el registro manual, que es propenso a errores y genera una pérdida de tiempo, errores en el cálculo de bonos y descuentos, absentismo laboral y descontento en los pagos.

En el ámbito educativo, la falta de precisión en el registro de asistencia puede traducirse en problemas para evaluar el compromiso y desempeño de los estudiantes, lo que a su vez afecta negativamente la calidad de la enseñanza y la experiencia académica. Según Shireesha y otros autores (2013), los sistemas tradicionales de asistencia son insuficientes para satisfacer las demandas de instituciones modernas, especialmente en contextos donde se busca optimizar recursos y mejorar la exactitud de los datos [**b16**].

La falta de precisión en el control de asistencia puede afectar negativamente tanto la calidad de la enseñanza como la experiencia de los estudiantes, lo que subraya la necesidad de un enfoque más eficiente.

3. Pregunta de investigación.

¿Cómo se puede implementar un sistema de automatización que utilice reconocimiento facial y de caracteres para mejorar la precisión en la gestión de asistencia en instituciones educativas y laborales?

4. Justificación.

La gestión de asistencia es un proceso crucial en diversos contextos, como instituciones educativas y empresas, donde es fundamental tener un control preciso y eficiente sobre el personal o los participantes. Sin embargo, los métodos tradicionales utilizados para el registro de asistencia, como las listas manuales o las credenciales de identificación, presentan varias desventajas. Estos enfoques son susceptibles a errores humanos, pueden ser objeto de manipulación, son ineficientes en términos de tiempo y requieren una intervención constante por parte del personal para asegurar su correcta aplicación.

En este contexto, el reconocimiento facial ha demostrado ser una herramienta eficaz para la identificación automática de personas, evitando los posibles errores manuales y mejorando la precisión en el control de asistencia.

Este proyecto se presenta como una herramienta que podría ser altamente efectiva para optimizar los sistemas de control de asistencia, especialmente en un contexto donde la digitalización de procesos es cada vez más relevante. La combinación de reconocimiento facial y reconocimiento de caracteres ofrece una solución moderna que no solo podría agilizar el registro de asistencia, sino que también tiene el potencial de reducir errores y fraudes, incrementar la productividad y mejorar la experiencia de los usuarios.

5. Objetivos.

5.1. Objetivo general.

Desarrollar e implementar un sistema de automatización para la gestión de asistencia en instituciones que utilice tecnologías de reconocimiento facial y de caracteres.

5.2. Objetivos específicos.

- Investigar y seleccionar las tecnologías adecuadas de reconocimiento facial y de caracteres que se integren en el sistema propuesto.
- Diseñar un prototipo funcional del sistema de gestión de asistencia que incluya una interfaz de usuario.
- Implementar el sistema para evaluar su rendimiento en términos de precisión y eficiencia en comparación con los métodos tradicionales de registro de asistencia.

6. Marco Referencial.

El problema que buscamos abordar ha persistido durante varios años, motivando la realización de múltiples iniciativas orientadas a su resolución. Diversos estudios han explorado la implementación de métodos y enfoques innovadores para monitorear la asistencia de manera eficiente.

Por ejemplo, Shoewu, Oluwagbemiga e Idowu (2012) propusieron un sistema de gestión de asistencia basado en datos biométricos. Este sistema compara los datos ingresados con los almacenados previamente en una base de datos y consta de dos etapas: **inscripción** y **autenticación**. Durante la inscripción, los datos biométricos se registran y almacenan en la base de datos principal, mientras que, en la autenticación, los datos se capturan nuevamente para compararlos con los existentes [**b4**].

Asimismo, Singh, Khan, Singh y otros (2015) propusieron un sistema compuesto por un hardware compacto, un servidor remoto y componentes de software. Este sistema permite la adquisición de datos manualmente o mediante sensores electrónicos. Además, el modelo puede ser adaptado para diversas aplicaciones, como encuestas, monitoreo de circuitos cerrados en industrias y hospitales, o sistemas de gestión de asistencia en escuelas y universidades [**b5**].

Por su parte, Kim (2016) desarrolló un prototipo para gestionar la asistencia utilizando **Near Field Communication (NFC)**, una tecnología de comunicación inalámbrica de corto alcance que permite la transferencia de datos entre dispositivos cercanos. El sistema propuesto por Kim consta de dos aplicaciones: una para profesores y otra para estudiantes. Este sistema automatiza la administración de la asistencia, resolviendo problemas como el registro por delegación, la pérdida de tiempo en clase y los costos adicionales asociados [**b6**].

En 2021, Kodali y Hemadri presentaron un método basado en reconocimiento facial para regis-

trar asistencia. Su enfoque utiliza una red supervisada de tamaño reducido para identificar rostros en un aula. Además, desarrollaron una aplicación web para facilitar el uso del sistema por parte de los usuarios [**b1**].

El diseño de sistemas de gestión de asistencia se fundamenta en diversos conceptos y tecnologías. En este proyecto, se emplearon herramientas como el **Reconocimiento Óptico de Caracteres (OCR)** y el algoritmo **Haar Cascade**.

El **OCR** permite convertir texto en imágenes escaneadas a datos procesables por computadora, independientemente de si el texto está escrito a mano o impreso. Su eficacia depende del preprocesamiento aplicado a las imágenes y, en muchos casos, del postprocesamiento para eliminar ruido y corregir errores [**b9**].

Por otro lado, el **Haar Cascade** es una técnica especializada en la detección de objetos mediante patrones rectangulares que evalúan cambios en el brillo de la imagen. Este método se utiliza ampliamente para identificar características específicas, como bordes o estructuras faciales, y se implementa mediante clasificadores en cascada, que optimizan la detección rápida y eficiente de objetos [**b10**].

Un concepto fundamental en estas tecnologías es la **Región de Interés (ROI)**, que delimita una parte específica de la imagen a manipular o analizar. Dependiendo del problema, esta región puede tener formas como rectángulos, círculos u otras configuraciones [**b11**]..

En este proyecto, se utilizó la biblioteca **Pytesseract** para el reconocimiento de texto, empleando la técnica de **momentos de Hu**. Este enfoque permite extraer siete características invariantes a transformaciones como traslación, escalado y rotación, facilitando la detección de objetos sin importar su posición en la imagen [**b14**].

7. Metodología.

El desarrollo del sistema de automatización para la gestión de asistencia mediante tecnologías de reconocimiento facial y de caracteres se llevará a cabo utilizando herramientas como OpenCV, redes neuronales, Python, Pytesseract y la biblioteca re. El proceso metodológico se estructurará en las siguientes fases:

7.1. Instalación de bibliotecas.

Las bibliotecas utilizadas para el proyecto fueron las siguientes:

1. tkinder
2. OpenCV
3. numpy

4. joblib
5. pyttsx3
6. time
7. datetime
8. collections
9. pyteseract

Aunque la mayoría de las bibliotecas vienen preinstaladas con Python en Visual Studio Code, en el caso de no ser así, el proceso para instalarla sería de la siguiente forma:

1. Abrir una terminal en visual studio code.
2. Escribe el comando pip install y con el nombre de la bibloteca correspondiente

No obstante, en el caso de Pyteseract es diferente ya que no se instala de la misma manera.

1. Instalar Python en el sistema (si aún no está instalado). Puedes descargarlo desde la página oficial: https://www.python.org/.
2. Instalar el paquete `pytesseract` utilizando `pip`. Abre una terminal o consola y ejecuta el siguiente comando:

```
pip install pytesseract
```

3. Descargar e instalar Tesseract-OCR, ya que `pytesseract` es simplemente una interfaz de Python para este software. Sigue estos pasos:
   a) Ve al repositorio oficial de Tesseract-OCR: https://github.com/tesseract-ocr/tesseract.
   b) Descarga la versión correspondiente a tu sistema operativo.
   c) Instala Tesseract siguiendo las instrucciones específicas de tu sistema:
      - **Windows**: Descarga el instalador ejecutable (EXE) y sigue el asistente de instalación.
      - **Linux**: Utiliza el siguiente comando:

```
sudo apt-get install tesseract-ocr
```

      - **MacOS**: Usa `brew` para instalarlo:

```
brew install tesseract
```

7.2. Fase de Recolección y Preparación de Datos.

En esta fase, se desarrolló una herramienta para extraer el nombre de la persona que sostendrá una tarjeta aproximadamente a la altura del pecho. El proceso para obtener y procesar los datos se llevó a cabo de la siguiente manera:

- Identificación de la región de interés (ROI): Se delimitó la región central de la imagen, donde se encuentra el texto (nombre de la persona), con el objetivo de crear un directorio de imágenes correspondiente a cada individuo.
- Preprocesamiento de la imagen: Se ajusta el tamaño de la ROI para ampliarla y se aplica una binarización invertida, mejorando la calidad del texto a extraer.
- Extracción de texto: Se utiliza la función de Pytesseract para convertir el contenido procesado en texto manejable.
- Limpieza de datos: Se emplea la biblioteca **'re'** para eliminar ruidos o caracteres no deseados en el texto, garantizando la correcta creación de los directorios.

Este proceso permite estructurar de manera organizada los datos en directorios etiquetados con el nombre de cada persona, lo que facilitará las predicciones posteriores [**13**].

Asimismo, estas mismas imágenes con el mismo procesamiento se van a utilizar para las predicciones del sistema de asistencia; la única diferencia será que la ROI estará en región central alta de la imagen para poder detectar el rostro.

Para ello, se desarrolló una función que busca los directorios dentro de una carpeta(la cual en el caso de este proyecto se llama test) para obtener no solo los nombres sino que la etiqueta de la clase para poder crear la base de datos en formato csv.

7.3. Fase de Procesamiento de Imágenes.

En esta etapa, se centró el esfuerzo en optimizar el procesamiento de las imágenes para extraer de manera eficiente las características relevantes. Para ello:

- Se utilizaron los momentos de Hu, que generan siete características invariantes ante transformaciones como traslación, rotación y escalado.

El resultado de este proceso es una base de datos estructurada con las características esenciales de las imágenes, la cual se empleó como entrada para el modelo propuesto (red neuronal).

7.4. Fase de Desarrollo y Entrenamiento del Modelo de Redes Neuronales.

En esta fase, se entrenó un modelo (red neuronal) utilizando las bases de datos generadas en la fase anterior. El modelo será diseñado para realizar predicciones basadas en siete posibles salidas, correspondientes a los nombres:

- Isabella (0)
- Joshua (1)
- Melissa (2)
- Zain (3)
- Angel (4)
- Jorge (5)
- Cristopher (6)

La implementación del modelo se realizó con la biblioteca Sklearn de Python, optimizando su estructura para garantizar un aprendizaje efectivo y resultados precisos. Además, se utilizó la biblioteca **'joblib'** para guardar tanto el modelo entrenado como la estandarización de los datos. Esto se decidió debido a la sensibilidad que tienen las redes neuronales a la escala de características. Adicionalmente, cada clase tiene un total de 50 imágenes, por lo que daría un total de 350 ejemplos para la base de datos.

7.5. Desarrollo de la aplicacion principal.

Con la base de datos y el modelo entrenado, se dio paso al desarrollo del pase de lista principal apoyado de bibliotecas como Tkinder de Python. Esto para la visualización de una terminal en pantalla la cual consiste en una pequeña ventana para iniciar el pase de lista o salir de la aplicación.

El primer paso fue desarrollar las funciones principales para los botones de la terminal y algunas complementarias de las mismas. A continuación, se describirán todas las funciones desarrolladas con su propósito.

1. **get file name()**: Función utilizada para crear el archivo '.txt' con el nombre de la ficha de hoy apoyándose de la biblioteca **'datetime'**. Su propósito es para crear archivos únicos del día donde se paso la lista.
2. **savemessagetofile()**: Función para guardar el nombre detectado por el alumno junto con la hora de llegada en el '.txt'.
3. **speaktext()**: Función utilizada para leer el mensaje generado después de la predicción junto con la hora llegada y que una voz lo repita. Para esto, se utilizo la biblioteca pyttsx3.

4. **calculateHuMoments()**: Función utilizada para transformar el frame de la cámara en un vector con la estructura de los ejemplos de la red neuronal para poder realizar la predicción.
5. **getRoiFaces()**: Función utilizada para obtener la región de interés del Frame del video.
6. **faceimagepreprocess()**: Función principal para transformar el frame de la imagen. Se le aplica el preprocesamiento descrito anteriormente para después obtener su región de interés con **'getRoiFaces'** y transformarlo con **'calculateHuMoments()'** para así tener el ejemplo con las características de la base de datos.
7. **showresultwindow**: Función principal donde se guarda el mensaje en un archivo .'.txt' después de mostrarlo en una ventana después de la predicción realizada con la función **'startcamera()'**, esta función tiene el fin principal de guardar la predicción en un '.txt' después de terminar con el calculo de la misma utilizando la función **'get file name()'** para crear o detractar el '.txt' y **'savemessagetofile()'** para guardarlo.
8. **startcamera()**: Función principal de la aplicación, se encarga de hacer la predicción del alumno mediante el uso de la cámara y el modelo entrenado. La lógica consiste en cargar el modelo y la estandarización de los datos para después abrir la cámara para detectar frames durante 10 segundos donde cada predicción por frame será guardado en un arreglo llamado **'predictions'**. Posteriormente se utilizó **'Counter'** de la biblioteca **'Collections'** se encuentra el nombre que mas veces apareció, con esto se determina al alumno detectado. Con este se crea el mensaje: **"Buenos días (clase - predicción). Hora de llegada: (hora actual - zona horario México)"**, para que finalmente sea mencionado por la función **'speaktext()'** y guardado en el '.txt' utilizando la función **showresultwindow()** la cual tras cerrarse regresa al menú principal de la aplicación.

7.6. Fase de Evaluación y Validación.

Una vez entrenado el modelo, se realizó a su evaluación y validación mediante métricas de rendimiento que comparen las predicciones de la red con los resultados esperados. Las métricas incluyeron:

- Precisión: Proporción de predicciones correctas sobre el total de predicciones realizadas.
- Tasa de error: Proporción de predicciones incorrectas respecto al total.

Para una mejor visibilidad, se utilizó una matriz de confusión de las clases predichas.

8. Resultados.

El sistema diseñado e implementado demostró ser capaz de capturar imágenes desde una cámara en tiempo real, procesarlas para extraer texto utilizando técnicas de preprocesamiento y OCR, y, posteriormente, identificar subdirectorios y almacenar imágenes según el texto extraído. A continuación, se describen los resultados obtenidos durante la ejecución del sistema con respecto a la detección del texto:

8.1. Detección de texto y extracción de nombres.

Mediante el análisis de las imágenes proporcionadas, se observó que el sistema pudo identificar correctamente regiones de interés y aplicar preprocesamiento (conversión a escala de grises, ampliación, inversión de colores y binarización) para facilitar la extracción del texto mediante pytesseract.

- Imagen 1: Imagen original obtenida aparir de la cámara digital.
- Imagen 2: Imagen de la selección de la región de interés para la búsqueda de texto.
- Imagen 3: Imagen bina rizada para la búsqueda de texto/identificación del nombre del gafete.

Los nombres extraídos fueron utilizados exitosamente para identificar subdirectorios, demostrando la capacidad del sistema para interpretar información textual desde imágenes capturadas.

8.2. Tasa de éxito en la extracción de texto.

La tasa de éxito en la extracción de texto dependió en gran medida de la calidad de la imagen y del contraste del texto con el fondo. En las imágenes proporcionadas, el sistema logró identificar caracteres alfanuméricos con un nivel aceptable de precisión, aunque en algunos casos presentó dificultades para distinguir caracteres pequeños o poco definidos.

Esto es debido al ruido existente en las imágenes en tiempo real, por la cual se debe realizar un preprocesamiento para la reducción del mismo y mejorar la precisión.

8.3. Visualización de los resultados.

En las pruebas realizadas, se obtuvieron imágenes procesadas que mostraban las regiones detectadas por MSER, indicando que el sistema logró identificar zonas con texto de manera efectiva. Estas zonas fueron fundamentales para optimizar la precisión del OCR en imágenes complejas.

8.4. Imágenes procesadas.

A continuación, se incluyen las imágenes capturadas y los resultados obtenidos del sistema:

Imagen 1: Imagen original obtenida mediante una cámara digital.

Figura 1: Fotografía de la alumna Melissa sosteniendo su gafete.

Imagen 2: Imagen de la selección de la región de interés para la búsqueda de texto.

Figura 2: Reducción de la imagen original de acuerdo al gafete.

Imagen 3: Imagen binarizada para la búsqueda de texto/identificación del nombre del gafete.

Figura 3: Conversión de la zona delimitada a binario.

Nota: Las imágenes muestran el preprocesamiento aplicado, incluyendo la segmentación del texto y las operaciones binarizadas utilizadas para facilitar el reconocimiento. Asimismo, para más observaciones de los resultados ir al apartado **Imagenes adicionales de los resultados** en el apartado de Anexos.

8.5. Resultados de la predicción con reconocimiento facial.

Durante el entrenamiento, el modelo alcanzó una precisión de predicción de 0.9578 (95.78 por ciento). Sin embargo, en la práctica, este rendimiento no siempre se mantiene.

La matriz de confusión obtenida durante las pruebas refleja el desempeño del modelo al clasificar correctamente las etiquetas de los individuos reconocidos (Isabella, Joshua, Melissa, Zain, Angel, Jorge y Cristopher). Los resultados muestran lo siguiente:

- Isabella (Clase 0): El modelo identificó correctamente 11 de 11 imágenes, demostrando una precisión perfecta para esta clase.
- Joshua (Clase 1): Se reconocieron correctamente 13 imágenes. No se observaron falsos positivos ni negativos, lo que indica una alta confiabilidad en esta clase.
- Melissa (Clase 2): Aunque 7 imágenes fueron clasificadas correctamente, una fue erróneamente asignada a la clase 6 (Cristopher). Esto sugiere cierta confusión entre estas dos clases.
- Zain (Clase 3): El modelo clasificó correctamente 12 imágenes, pero 1 fue asignada incorrectamente a la clase 6, mostrando un patrón similar al observado en la clase 2.
- Angel (Clase 4): Todas las imágenes de esta clase fueron clasificadas correctamente, indicando una precisión perfecta.
- Jorge (Clase 5): Se clasificaron correctamente 8 imágenes sin errores.
- Cristopher (Clase 6): Aunque 8 imágenes fueron correctamente reconocidas, 1 imagen de la clase 2 y otra de la clase 3 fueron clasificadas erróneamente como Cristopher, evidenciando cierta confusión con estas clases.

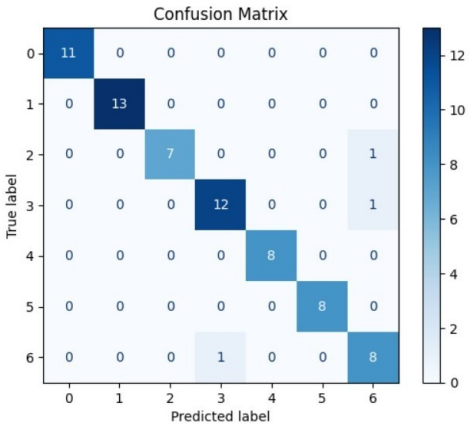

Figura 4: Matriz de confusión de los resultados obtenidos.

Una de las problemáticas identificadas es la confusión entre Isabella y Melissa, ya que el modelo tiende a predecir con mayor frecuencia a Melissa. Esto probablemente se debe a que, durante la binarización, el cabello de Isabella se convirtió en negro, lo que pudo llevar al modelo a asociarlo incorrectamente con Melissa.

En cuanto a las demás clases, el algoritmo muestra dependencia de la altura de las personas. Por ejemplo, Joshua es la clase que el modelo detecta con mayor facilidad. Además, las imágenes de Ángel presentaron una corrupción en el momento de su captura, lo que paradójicamente facilitó la identificación de su clase.

Figura 5: Proceso de identificación

No obstante, el modelo presenta limitaciones al depender de encontrarse a la distancia correcta de la cámara para lograr una buena predicción. En casos donde esto no se cumple, el modelo tiende a identificar incorrectamente a la persona como Isabella o Melissa, incluso si ninguna de ellas está presente. Además, uno de los problemas detectados radica en la calidad de la cámara, que puede

afectar significativamente las predicciones, como se observó durante los experimentos realizados en clase.

Adicionalmente, se tomó la decisión de que el programa con interfaz incluyera únicamente el modelo entrenado, sin la posibilidad temporal de añadir nuevas imágenes a la base de datos. Esta decisión se debe a la alta carga computacional que representa realizar ambos procesos de manera simultánea, lo cual generaba retrasos en la función encargada de leer el texto en voz.

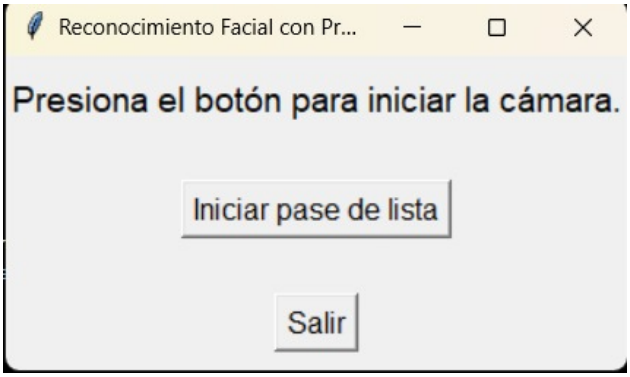

Figura 6: Interfaz gráfica del programa de pase de lista.

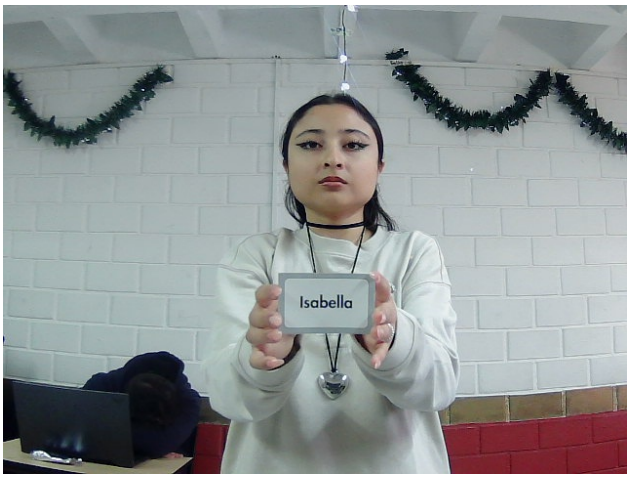

Figura 7: Proceso de predicción.

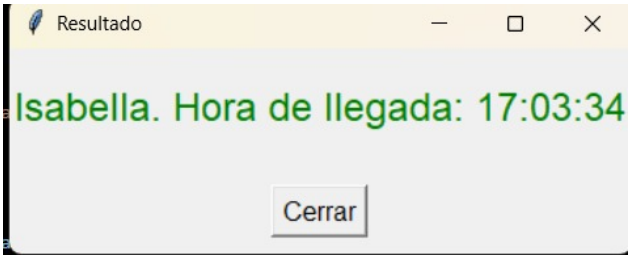

Figura 8: Mensaje de salida.

Por último, consideramos que una prueba interesante sería construir la base de datos a lo largo de un ciclo escolar o semestre. Esto permitiría agregar mayor variedad a las imágenes, ya que las personas tienden a cambiar con el tiempo. Analizar si esta estrategia mejora o perjudica las predicciones podría aportar valiosas conclusiones al desarrollo del modelo.

9. Resultados del salón de clases

Durante la prueba realizada durante el examen nos dimos cuenta de que nuestro modelo no predecía correctamente, solo predecía a Melissa y los demás intentos fueron erróneos, teniendo un total de 2/11 ejemplos bien predichos.

Esto nos llevó a revisar el modelo y nos percatamos de que el problema fue durante el entrenamiento del mismo; originalmente, se decidió separar la base de datos en 80 % para train y 20 % para test, sin embargo, se ajustó a 70 % y 30 % respectivamente y fue así que el modelo ya pudo predecir correctamente al alumno, incluso teniendo menos precisión que el otro al tener 93.89 %.

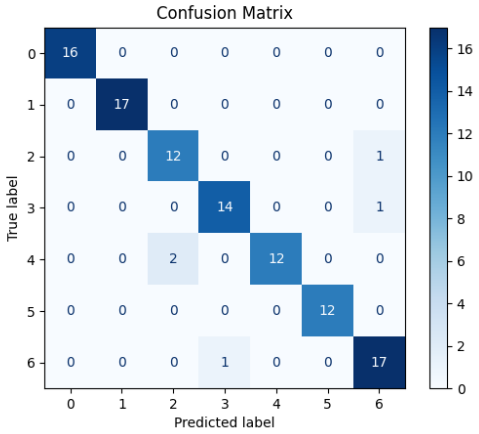

Figura 9: Matriz de confusión del segundo modelo con 70 % para el entrenamiento y 30 % para la prueba

Adicionalmente, se le agregó una pausa de 0.5 segundos utilizando la función time.sleep en la obtención de los frames, esto ayudó más a la prueba debido a que capturaba menos frames y, por ende, aumentaba la calidad de los frames y junto al modelo, pudieron predecir correctamente ya que no castigaba tanto el momento de cuando se equivocaba en la predicción.

```
Melissa. Hora de llegada: 08:12:24
Isabella. Hora de llegada: 08:16:37
Melissa. Hora de llegada: 08:17:10
Joshua. Hora de llegada: 08:17:40
Joshua. Hora de llegada: 08:18:58
Joshua. Hora de llegada: 08:21:14
Isabella. Hora de llegada: 08:24:23
Melissa. Hora de llegada: 08:26:36
Angel. Hora de llegada: 08:27:22
Joshua. Hora de llegada: 08:28:16
Isabella. Hora de llegada: 08:28:44
Melissa. Hora de llegada: 08:34:00
Angel. Hora de llegada: 08:39:37
Angel. Hora de llegada: 08:45:33
Angel. Hora de llegada: 08:48:49
Angel. Hora de llegada: 08:50:30
Joshua. Hora de llegada: 08:51:11
Joshua. Hora de llegada: 08:51:45
Angel. Hora de llegada: 08:52:17
Jorge. Hora de llegada: 10:08:03
```

Figura 10: Salida de las pruebas con el modelo original

```
Joshua. Hora de llegada: 10:08:40
Isabella. Hora de llegada: 10:09:31
Melissa. Hora de llegada: 10:10:00
```

Figura 11: Salida de las pruebas con el segundo modelo

10. Conclusión.

El sistema demostró ser funcional para la captura, procesamiento y almacenamiento de imágenes, integrando exitosamente tecnologías como OCR y la generación dinámica de directorios. No obstante, factores como la iluminación, la calidad del texto y el ruido visual pueden influir en su desempeño, lo que señala áreas de mejora. En cuanto a la predicción de rostros, uno de los principales desafíos fue la distancia de la persona a la cámara, una problemática que requirió ajustes diarios durante las sesiones en clase. A pesar de ello, el sistema logró desenvolverse de manera satisfactoria.

Adicionalmente, resolver el problema del segundo modelo nos ayudó a comprender que no siempre debemos confiar en una buena precisión durante el entrenamiento, sobre todo en los problemas de visión donde el ruido en las nuevas entradas puede llegar a afectar la salida.Por lo que, nos llevamos como experiencia el comparar la predicción con el resultado en práctica para así identificar y realizar los ajustes necesarios.

Trabajar con visión artificial resultó ser una experiencia muy interesante, pues nos permitió

explorar las diversas formas en las que los desarrolladores pueden enviar información a la computadora. Diseñar métodos para transmitir esta información de la manera más efectiva fue un reto sumamente enriquecedor de resolver.

11. Trabajos futuros.

A futuro, se planea implementar un sensor que detecte automáticamente si una persona se encuentra cerca, eliminando la necesidad de presionar un botón para el pase de lista. Además, se contempla el uso de un sistema de gestión de bases de datos, como MySQL, para el almacenamiento eficiente de la información. También se buscará que el sistema pueda detectar automáticamente la materia en curso. Esto, mediante la adición de un horario escolar, optimizando aún más su funcionalidad.

Referencias

[1] Kodali, R. K., & Hemadri, R. V. (2021). Attendance management system. In *2021 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-5). IEEE.

[2] Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*, 20-21.

[3] García, I., & Caranqui, V. (2015). La visión artificial y los campos de aplicación. *Tierra infinita*, 1(1), 98-108.

[4] Shoewu, O., & Idowu, O. A. (2012). Development of attendance management system using biometrics. *The Pacific Journal of Science and Technology*, 13(1), 300-307.

[5] Singh, M., Khan, M. A., Singh, V., Patil, A., Wadar, S., et al. (2015). Attendance management system. In *2015 2nd International Conference on Electronics and Communication Systems (ICECS)* (pp. 418-422). IEEE.

[6] Kim, B.-G. (2016). An implementation of auto attendance management system based on app using NFC technique. *Journal of the Korea Academia-Industrial cooperation Society*, 17(2), 719-723.

[7] Penalva, J. (2024, May). NFC: qué es y para qué sirve en este 2024. *Xataka*. Retrieved from https://www.xataka.com/basics/nfc-que-es-y-para-que-sirve.

[8] Romero, J. (2021, September). ¿Qué es el NFC y para qué sirve?. *Geeknetic*. Retrieved from https://www.geeknetic.es/NFC/que-es-y-para-que-sirve.

[9] R. Smith, ."An Overview of the Tesseract OCR Engine,"2007 Ninth International Conference on Document Analysis and Recognition (ICDAR), Curitiba, Brazil, 2007, pp. 629-633, doi: 10.1109/ICDAR.2007.4376991.

[10] R. Padilla, C. F. F. Costa Filho, and M. G. F. Costa, ."Evaluation of Haar cascade classifiers designed for face detection,"International Journal of . . . , 2012. [Online]. Available: Academia.edu.

[11] S. Soo, ."Object detection using Haar-cascade Classifier,"Institute of Computer Science, University of Tartu, 2014. [Online]. Available: Academia.edu.

[12] R. N. Chityala and K. R. Hoffmann, Region of Interest (ROI) computed tomography,"Physics of . . . , 2004. [Online]. Available: spiedigitallibrary.org.

[13] re — Regular expression operations,"Python Documentation, n.d. [Online]. Available: https://docs.python.org/es/3/library/re.html.

[14] Hu, M. K. (1962). Visual pattern recognition by moment invariants. IRE transactions on information theory, 8(2), 179-187.

[15] V. Patel, N. Shah, and R. Patel, ."Automated Attendance Management System Based on Face Recognition Algorithms,"Proceedings of the IEEE International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018. [Online]. Available: https://doi.org/10.1109/ICCCNT.2018.8494045.

[16] Chintalapati, S., Raghunadh, M. (2013). Automated attendance management system based on face recognition algorithms. En 2013 International Conference on Computational Intelligence and Computing Research (ICCIC) (pp. 1-5). IEEE. doi: 10.1109/ICCIC.2013.6724266.

12. Anexos.

12.1. Código fuente.

```
import tkinter as tk
from tkinter import messagebox
import cv2
import numpy as np
import joblib
import pyttsx3
import time
import datetime
from collections import Counter

# Inicializar pyttsx3
engine = pyttsx3.init()

def get_file_name():
    today = datetime.datetime.now().strftime("%Y-%m-%d")
    file_name = f"{today}.txt"
    return file_name

def save_message_to_file(message):
    file_name = get_file_name()  # Obtener el nombre del archivo
    try:
        with open(file_name, "a") as file:
            file.write(f"{message}\n")  # A adir el mensaje con un
               salto de l nea
        print(f"Mensaje guardado en {file_name}: {message}")
    except Exception as e:
        print(f"Error al guardar el mensaje: {e}")

def speak_text(text):
    engine.setProperty('rate', 150)
    engine.setProperty('volume', 0.9)
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[0].id)

    engine.say(text)
    engine.runAndWait()

def calculateHuMoments(img):
    moments = cv2.moments(img)
    hu_moments = cv2.HuMoments(moments).flatten()  # Convertir a un
       array plano
    return hu_moments

```

```
42 def getRoi_Faces(img, reduction_factor=0.5):
43     height, width = img.shape[:2]
44     roi_height = height // 2
45     reduced_width = int(width * reduction_factor)
46     x_start = (width - reduced_width) // 2
47     y_start = 0
48     x_end = x_start + reduced_width
49     y_end = roi_height
50     roi = img[y_start:y_end, x_start:x_end]
51     return roi
52 
53 def face_image_preprocess(img):
54     gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
55     _, binary = cv2.threshold(gray_image, 128, 255, cv2.
           THRESH_BINARY)
56     roi = getRoi_Faces(binary)
57     hu_moments = calculateHuMoments(roi)
58     reshaped_array = np.array(hu_moments.tolist()).reshape(1, -1)
59     return reshaped_array
60 
61 def start_camera():
62     cap = cv2.VideoCapture(0)
63     if not cap.isOpened():
64         messagebox.showerror("Error", "Error al abrir la c mara.")
65         return
66 
67     # Cargar el modelo y el escalador
68     try:
69         mlp_loaded = joblib.load('./mlp_mode_Paselista123.pkl')
70         scaler_loaded = joblib.load('./scaler_paselista2.pkl')
71     except Exception as e:
72         messagebox.showerror("Error", f"Error al cargar el modelo: {
               e}")
73         return
74 
75     start_time = time.time()
76     predictions = []
77 
78     while True:
79         ret, frame = cap.read()
80         if not ret:
81             print("Error al capturar el frame. Verifica que la
                   c mara est  disponible.")
82             break
83 
84         # Procesar imagen y predecir
85         face = face_image_preprocess(frame)
```

```
86          new_example_scaled = scaler_loaded.transform(face)
87          prediction = mlp_loaded.predict(new_example_scaled)[0]
88          print(f"Prediction ->>>>>>>>>>>>>>>>>>>>>>>>>>>{prediction}")
89          predictions.append(prediction)
90          time.sleep(0.5)
91
92          # Mostrar la c mara
93          cv2.putText(frame, "Identificando...", (50, 50),
94                      cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 0), 2, cv2.
                        LINE_AA)
95          cv2.imshow('Camera', frame)
96
97          # Salir si se presiona 'q' o si pasan 10 segundos
98          if cv2.waitKey(1) & 0xFF == ord('q') or time.time() -
                start_time > 10:
99              break
100
101     cap.release()
102     cv2.destroyAllWindows()
103
104     # Determinar la predicci n m s frecuente
105     if predictions:
106         most_common_prediction = Counter(predictions).most_common(1)
                [0][0]
107         current_time = datetime.datetime.now().strftime("%H:%M:%S")
108         final_message = f"                                    Buenos
                d as {most_common_prediction}. Hora de llegada: {
                current_time}"
109         speak_text(final_message)
110         show_result_window(f"{most_common_prediction}. Hora de
                llegada: {current_time}")
111     else:
112         messagebox.showinfo("Informaci n", "No se realizaron
                predicciones.")
113
114 def show_result_window(message):
115     result_window = tk.Toplevel()
116     result_window.title("Resultado")
117
118     # Mostrar el mensaje en la ventana
119     result_label = tk.Label(result_window, text=message, font=("
            Arial", 18), fg="green")
120     result_label.pack(pady=20)
121
122     # Guardar el mensaje en un archivo de texto autom ticamente
123     save_message_to_file(message)
124
```

```
    # Funci n para cerrar la ventana despu s de 5 segundos
    def close_after_delay():
        result_window.destroy()

    # Configurar el temporizador para cerrar la ventana despu s de
       5 segundos
    result_window.after(3000, close_after_delay)

    # Agregar el bot n de cerrar manualmente
    close_button = tk.Button(result_window, text="Cerrar", command=
       result_window.destroy, font=("Arial", 12))
    close_button.pack(pady=10)

# Interfaz gr fica con tkinter
ventana = tk.Tk()
ventana.title("Reconocimiento Facial con Predicciones")

# Elementos de la interfaz
label_var = tk.StringVar()
label_var.set("Presiona el bot n para iniciar la c mara.")

label = tk.Label(ventana, textvariable=label_var, font=("Arial", 14)
   )
label.pack(pady=10)

boton_iniciar = tk.Button(ventana, text="Iniciar pase de lista",
   command=start_camera, font=("Arial", 12))
boton_iniciar.pack(pady=20)

boton_salir = tk.Button(ventana, text="Salir", command=ventana.
   destroy, font=("Arial", 12))
boton_salir.pack(pady=10)

ventana.mainloop()
```

Listing 1: Código del algoritmo en Python.

12.2. Imágenes adicionales del procedimiento.

A continuación, se presentan más imágenes adicionales que ilustran el desempeño del sistema en diversas etapas del proceso. Estas imágenes complementan los resultados mostrados anteriormente, ofreciendo una visión más detallada de cómo el sistema maneja las distintas fases de captura, procesamiento y análisis. Estas imágenes permiten observar con mayor claridad las interacciones del sistema con los usuarios y el entorno durante las pruebas realizadas.

Figura 12: Muestra de la región de interés, con la binarización invertida - Alumno Joshua.

Figura 13: Muestra de la región de interés, con la binarización invertida - Alumna Isabella.

Figura 14: Muestra de la región de interés, con la binarización invertida - Alumna Melissa.

Figura 15: Muestra de la región de interés, con la binarización invertida - Alumno Cristopher.

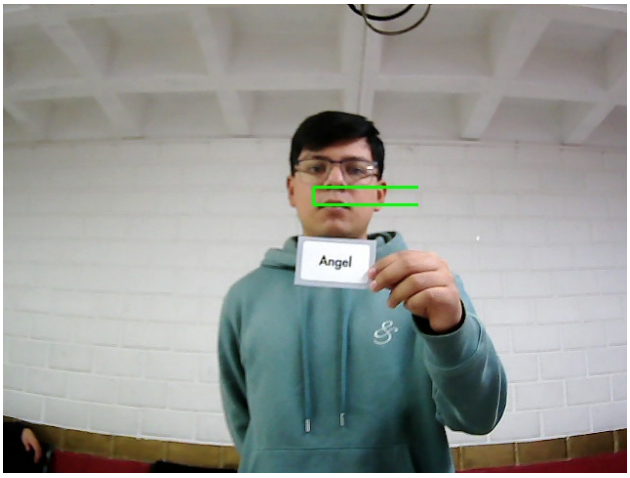

Figura 16: Muestra de la captura de la cámara, antes de la binarización invertida y selección de interés - Alumno Ángel.

Figura 17: Muestra de la región de interés, con la binarización invertida - Alumno Ángel.

Figura 18: Muestra de la región de interés, con la binarización invertida - Alumno Zain.