ISSN: 2594 - 2905

ENTRENAMIENTO DE UN MODELO PARA RECONOCIMIENTO FACIAL APLICADO A UN SISTEMA DE PRÉSTAMO DE EQUIPO

TRAINING OF A MODEL FOR FACIAL RECOGNITION APPLIED TO AN EQUIPMENT LOAN SYSTEM

Francisco Gibrán García Candelario[1], Natividad Juárez González [2], José-Aurelio Ramírez González [3], Wendy Carranza Díaz [4]

[1] Maestría en Ciencias Computacionales y Telecomunicaciones, Tecnológico Nacional de México, Campus Acayucan, Departamento de Ingeniería en Sistemas Computacionales, gibran.gc@acayucan.tecnm.mx , Carretera Costera del Golfo km. 216.4, Colonia Agrícola Michapan.

[2] Maestría en Tecnologías de la Información, Tecnológico Nacional de México, Campus Acayucan, Departamento de Ingeniería en Sistemas Computacionales, natividad.jg@acayucan.tecnm.mx, , Carretera Costera del Golfo km. 216.4, Colonia Agrícola Michapan.

[3] Maestría en Tecnología Educativa, Tecnológico Nacional de México, Campus Acayucan, Departamento de Ingeniería en Sistemas Computacionales, aurelio.rg@acayucan.tecnm.mx, Carretera Costera del Golfo km. 216.4, Colonia Agrícola Michapan.

[4] Maestría en Tecnología Educativa, Tecnológico Nacional de México, Campus Minatitlán, Departamento de Ingeniería en Sistemas Computacionales, wendy.cd@minatitlan.tecnm.mx

Resumen -- El reconocimiento facial abre una alternativa a los sistemas computacionales que requieren de autenticación de usuarios permitiendo su aplicación en seguridad de información, tarjetas inteligentes, control de acceso. Actualmente existen servicios que realizan el reconocimiento facial con un cargo monetario. En este artículo mostramos el entrenamiento de un modelo de visión artificial para realizar localmente el reconocimiento facial de usuarios que solicitan préstamo de equipo evitando el costo económico que otras aplicaciones generan. Se desarrolló una API que utiliza el modelo entrenado aplicando el algoritmo LBPH a través de la biblioteca de código abierto OpenCV, para determinar si un rostro se encuentra en el conjunto de rostros válido. La API es usada en una aplicación web que registra el préstamo de equipo para el Tecnológico Nacional de México campus Acayucan.

Se logró obtener una exactitud F1-Score del 90% sin depender de ninguna tecnología de pago o propietaria con un tiempo de predicción menor a los 100 milisegundos por consulta.

Palabras Clave: *Reconocimiento facial, LBPH, OpenCV, API.*

Abstract

Facial recognition opens an alternative to computer systems that require user authentication, allowing its application in information security, smart cards, and access control. Currently there are services that perform facial recognition with a monetary charge. In this article we show the training of an artificial vision model to locally perform facial recognition of users requesting equipment loan, avoiding the monetary charges that other applications request. An API was developed that uses the trained model applying the LBPH algorithm through the open source library OpenCV to determine if a face is found in the valid face set. The API is used in a web application that records the loan of equipment for the Tecnológico Nacional de México campus Acayucan.

It was possible to obtain an F1-Score accuracy of 90% without relying on any proprietary or payment technology with a prediction time of less than 100 milliseconds per query.

Key words: *Face recognition, LBPH, OpenCV, API.*

INTRODUCCIÓN

Antecedentes del problema

El proceso de reconocimiento facial consiste principalmente en comparar la información presentada en una base de datos, con los datos de prueba obtenidos de la persona que se desea reconocer. Para realizar esta comparación se pueden utilizar algoritmos que utilizan los valores de intensidad en escala de grises de los píxeles de la imagen. [1]

En el TecNM campus Acayucan se cuenta con un servicio de préstamo de equipo (videoproyectores, equipo de cómputo, accesorios electrónicos, etc). Inicialmente se escribía en una bitácora física los nombres y datos de las

iPSUMTEC
ISSN: 2594 - 2905

personas que solicitaban un equipo, lo que implicaba una inversión de tiempo de al menos un par de minutos. Posteriormente, se reemplazó por una aplicación web que realiza el registro, disminuyendo el tiempo de registro de datos cuando se solicitaba un equipo. Para validar al usuario se consideró utilizar un sistema de contraseñas, lo que actualmente no se considera muy eficiente.

Es sabido que un esquema convencional de contraseñas es vulnerable a ataques como *shoulder surfing* (mirar por encima del hombro), *key loggers* (registradores de teclas), *phishing*, y *login spoofing* (suplantación del inicio de sesión). [2]

Ahora bien, un sistema que utiliza reconocimiento facial tiene la ventaja de evitar el compartir contraseñas, evitando así la suplantación de identidad entre usuarios. La cantidad de tiempo requerida para realizar la autenticación es menor si se toma en cuenta el tiempo de tecleo de la contraseña, contra el solo hecho de pulsar un botón para capturar la imagen.

Comparados con otros sistemas biométricos, los sistemas de reconocimiento facial son no intrusivos, y cada vez son más aceptados por los usuarios [3], además del bajo costo del equipo de captura [4].

Estado actual del tema

Se dice que un sistema puede realizar detección facial cuando es capaz de detectar que un rostro humano está presente en una imagen o video. Existen dos enfoques principales para realizar la detección facial: los enfoques basados en características, y los enfoques basados en imágenes.

Entre los enfoques basados en características existen algoritmos que utilizan análisis de características. Estos algoritmos buscan encontrar características estructurales sin importar, entre otras, las variaciones en las condiciones de iluminación. Viola-Jones y AdaBoost son ejemplos de este enfoque. [5]

Por otra parte, el reconocimiento facial consiste en confirmar la identidad de un rostro encontrado.

El reconocimiento facial tiene hoy un alto rendimiento y se han resuelto la mayoría de los problemas que dificultan su uso. Sin embargo, aún no es posible evitar por completo los errores. [6]

Las técnicas de reconocimiento facial se pueden dividir también en las que se basan en características (como la geometría del rostro), y las que se basan en la representación del rostro completo. Éstas últimas se dividen en dos enfoques: el enfoque estadístico, y el enfoque que utiliza Inteligencia Artificial (IA). [4]

Entre los métodos que utilizan IA se encuentra Local Binary Pattern (LBP).

LBP asigna una etiqueta a cada píxel de la imagen estableciendo un umbral entre el píxel central y los píxeles vecinos, como se indica en la Ec(1).

$$f(I(Z_0), I(Z_i)) = \begin{cases} 0, \text{if} I(Z_i) - I(Z_0) \leq threshold \\ 1, \text{if} I(Z_i) - I(Z_0) > threshold \end{cases}, i = 1,2,\ldots,8$$

Ec(1)

Por ejemplo, considere que una imagen tuviera los valores que se muestran en la figura 1, que representan la intensidad en una escala de grises.

6	3	7
8	4	2
3	5	1

Figura 1. Ejemplo de valores de intensidad en una imagen.

El algoritmo LBP obtendría un micro patrón con los valores que se muestran en la figura 2.

1	0	1
1		0
0	1	0

Número binario: 10100101
Número decimal: 165

. Figura 2. Obtención de micropatrón.

LBP es fácil de calcular y ha sido usado exitosamente en reconocimiento facial. [7]. Comparado con otras técnicas como Haar Cascade, LBP es menos preciso, pero su tiempo de ejecución es mucho menor. [8]

LBPH es un algoritmo que combina LBP con HOG (Histograms of Oriented Gradients). Se utilizan los valores proporcionados por LBP y se forma un vector de datos que describen el patrón de la imagen original. Se crea entonces un histograma con la frecuencia de las

ocurrencias LBP, que son valores entre 0 y 255, por lo que el histograma contendrá 256 posiciones. [9]

LBP en combinación con HOG han logrado alcanzar un excelente rendimiento en la detección humana incluso con oclusión parcial. [10]

Otras ventajas de utilizar este algoritmo son: no es afectado por los cambios de iluminación, se puede re-entrenar, requiere poco preprocesamiento de las imágenes, y comparado con otros algoritmos de reconocimiento facial (Eigenfaces, Fisherfaces) tiene menores márgenes de error en reconocimiento. [11]

OpenCV es una biblioteca de código abierto para visión por computadora ampliamente utilizada por empresas, grupos de investigación y organismos gubernamentales. Cuenta con una comunidad grande de desarrolladores. Tiene su propio módulo de reconocimiento facial que implementa, entre otros, el algoritmo LBPH. [12]

REST es un protocolo simple basado en texto usado para que programas que se estén ejecutando en diferentes computadoras se comuniquen a través de Internet. Una API de REST es una interfaz de programación de aplicaciones que permite comunicar, mediar e interactuar con un sistema para obtener datos o ejecutar alguna función. [13]

Python es un lenguaje de programación que, por sus características, es una buena opción para científicos e ingenieros que escriben aplicaciones científicas. [14]. Python cuenta con una versión de OpenCV y con bibliotecas de funciones que permiten crear una API REST para comunicar un modelo de reconocimiento facial con aplicaciones externas.

Definición del problema

Se desea entrenar un modelo de visión artificial para reconocimiento facial de un conjunto de rostros. Debido a que la cantidad de usuarios aumenta con el tiempo, es necesario que el modelo entrenado pueda ser modificado posteriormente, agregando nuevos rostros a la base de datos.

Una vez obtenido el modelo entrenado, se requiere: a) medir su exactitud y precisión y b) probarlo en una API propia que reciba una entrada de datos (una imagen), y devuelva la clave o el nombre de la persona encontrada.

Se desea utilizar esta API en la aplicación web existente en la Institución, de manera que podamos enviar en forma asíncrona una imagen capturada a través de una cámara. La aplicación web recibirá el resultado que devuelva la API y continuará con sus operaciones relacionando la operación que se está efectuando con la persona detectada.

Objetivo general

Entrenar un modelo para reconocimiento facial para su uso en una aplicación web de préstamo de equipo.

Objetivos específicos

- Entrenar un modelo para reconocimiento facial utilizando el algoritmo LBPH.
- Evaluar el modelo entrenado midiendo su exactitud, precisión y F1-Score.
- Crear una API que utilice el modelo entrenado para predecir el nombre o clave de una persona, dada una imagen de su rostro.
- Realizar peticiones a la API desde una aplicación web de préstamo de equipo.
- Realizar un script para modificar el modelo entrenado y guardado a fin de incorporar nuevos rostros.

Justificación

Con el logro de los objetivos se conseguirá un ahorro para la institución. El uso de API's comerciales, por ejemplo AWS Rekognition, incurre en un costo de US $ 1.00 por mil imágenes procesadas mensualmente. Utilizando la API local se estarían ahorrando hasta MX $ 720 anuales considerando un uso de 100 imágenes procesadas diariamente. Otros servicios comerciales similares se muestran en la Tabla 1.

Tabla 1. Costos de procesamiento de imágenes por servicios comerciales.

Nombre del servicio	Costo por 1000 imágenes	Sitio web
Rekognition	US $ 1.00	https://aws.amazon.com/rekognition/
Cloud Vision	US $ 1.50	https://cloud.google.com/vision/
Face API	US $ 1.00	https://azure.microsoft.com/es-mx/services/cognitive-services/face/

Además del beneficio económico, el usuario tendrá una mejor experiencia de uso, pues se evita tener que teclear sus credenciales, lo que hace más largo el tiempo de espera para las demás personas.

Se incrementa la seguridad, evitando en cierta medida la suplantación o falsificación de identidad.

DESARROLLO

A continuación mostramos la metodología usada para desarrollar el proyecto, que consta de 3 grandes etapas:

1. Entrenamiento del modelo
2. Elaboración de la API
3. Implementación dentro de la aplicación web

Metodología

Para el desarrollo de cada una de las tres etapas ya mencionadas se realizaron las siguientes actividades.

Etapa de Entrenamiento del modelo

a) Adquirir un conjunto de imágenes del rostro de los usuarios del sistema
b) Elaborar el proceso de entrenamiento a todo el conjunto de imágenes y guardar el modelo entrenado.
c) Realizar una evaluación al modelo recién entrenado
d) Elaborar el proceso de entrenamiento a una sola imagen y modificar el modelo ya entrenado.
e) Realizar una evaluación al modelo guardado con modificaciones.
f) Realizar una prueba usando una imagen preseleccionada.

Etapa de Elaboración de la API

g) Seleccionar la librería para creación de la API
h) Utilizar el modelo entrenado dentro de la API
i) Extraer la imagen enviada a la API
j) Buscar un rostro dentro de la imagen y predecir el resultado a través del modelo entrenado
k) Enviar el resultado como respuesta, si está dentro de un umbral aceptable.

Etapa de Implementación dentro de la aplicación web

l) Acceder a la cámara del equipo
m) Tomar una fotografía cuando el usuario lo indique
n) Tratamiento de la imagen para poder ser enviada a través de protocolo HTTP
o) Envío de la imagen en forma asíncrona
p) Recepción del resultado y tratamiento

Se describe a continuación cada paso del proceso. Se utilizó el lenguaje de programación Python 3.8 en las etapas de Entrenamiento del Modelo y Elaboración de la API, y se usó el lenguaje de programación Javascript para la etapa de Implementación dentro de la aplicación web.

Entrenamiento del modelo

a) Adquirir un conjunto de imágenes del rostro de los usuarios del sistema

Se solicitó a cada persona que utilizará el sistema un conjunto de por lo menos 30 imágenes de su rostro. Las imágenes que cada persona proporcionó se guardan en una carpeta con un nombre descriptivo. Todas las carpetas se guardan en un mismo directorio.

b) Elaborar el proceso de entrenamiento a todo el conjunto de imágenes y guardar el modelo entrenado.

El algoritmo utilizado fue el siguiente:
1. Leer el directorio con las imágenes.
2. Crear un archivo de texto para guardar las etiquetas.
3.. Para cada carpeta de imágenes:
    3.1 Utilizar un número consecutivo como etiqueta y guardar éste en el archivo
    3.2 Para cada imagen dentro de la carpeta:
        3.2.1 Extraer el rostro detectado y guardarlo en un array.
        3.2.2. Guardar en un segundo array la etiqueta correspondiente.
4. Dividir el conjunto de datos obtenidos (arrays) en particiones de entrenamiento y prueba.
5. Usar LBPH para entrenar la partición con los datos de entrenamiento.

c) Realizar una evaluación al modelo recién entrenado

Se utilizan los datos de prueba para realizar un reporte del resultado de la clasificación realizada considerando las siguientes métricas: *precision*, *recall* y *f1-score*. También se calculó la métrica *accuracy*, aunque las primeras tres métricas tienen más importancia dado que se busca maximizar la cantidad de verdaderos positivos detectados.

Precisión es la proporción de rostros identificados correctamente del total de elementos identificados como positivos. Se calcula:

$$\frac{TP}{TP + FP}$$

IPSUMTEC
ISSN: 2594 - 2905

Recall es la proporción de los casos positivos que fueron predichos como positivos. Es decir:

$$\frac{TP}{TP+FN}$$

F1-Score es una media armónica que combina las medidas de *precision* y *recall* en un solo valor y se calcula:

$$2 * \frac{Preci i * Recall}{Preci i + Recall}$$

Accuracy es el porcentaje de rostros reconocidos correctamente, y se calcula:

$$\frac{TP + TN}{TP + FP + FN + TN}$$

donde:

TP y FP (True Positive y False Positive) se refieren a la cantidad de predicciones positivas que fueron correctas (TP) e incorrectas (FP).

TN y FN (True Negative y False Negative) se refieren a la cantidad de predicciones negativas que fueron correctas (TN) e incorrectas (FN). [15]

La evaluación fue realizada usando también el lenguaje de programación Python y su librería SkLearn.

d) Elaborar el proceso de entrenamiento a una sola imagen y modificar el modelo ya entrenado.

Una vez entrenado el modelo se desea tener un módulo que pueda agregar una nueva imagen para que pueda ser reconocida por el modelo.

El algoritmo utilizado es similar al algoritmo mostrado anteriormente, con la diferencia que se le proporciona como entrada la carpeta con los rostros de la persona a agregar.

e) Realizar una evaluación al modelo que fue modificado

Se realizó un script en Python que realice el proceso de evaluación a un modelo que haya sido modificado recientemente. Se elaboró de forma similar al proceso de evaluación antes descrito.

f) Realizar una prueba usando una imagen preseleccionada

Se realizaron pruebas usando imágenes preseleccionadas. El modelo arrojaba la imagen proporcionada y la etiquetaba con el nombre o clave de la persona reconocida.

Elaboración de la API

g) Seleccionar la librería para creación de la API y utilizarla.

Se utilizó la librería Hug para Python para implementar una interfaz REST (http://www.hug.rest/). La API abre un puerto para recibir peticiones POST a través de HTTP, especificando una ruta relacionada con una función dentro del código.

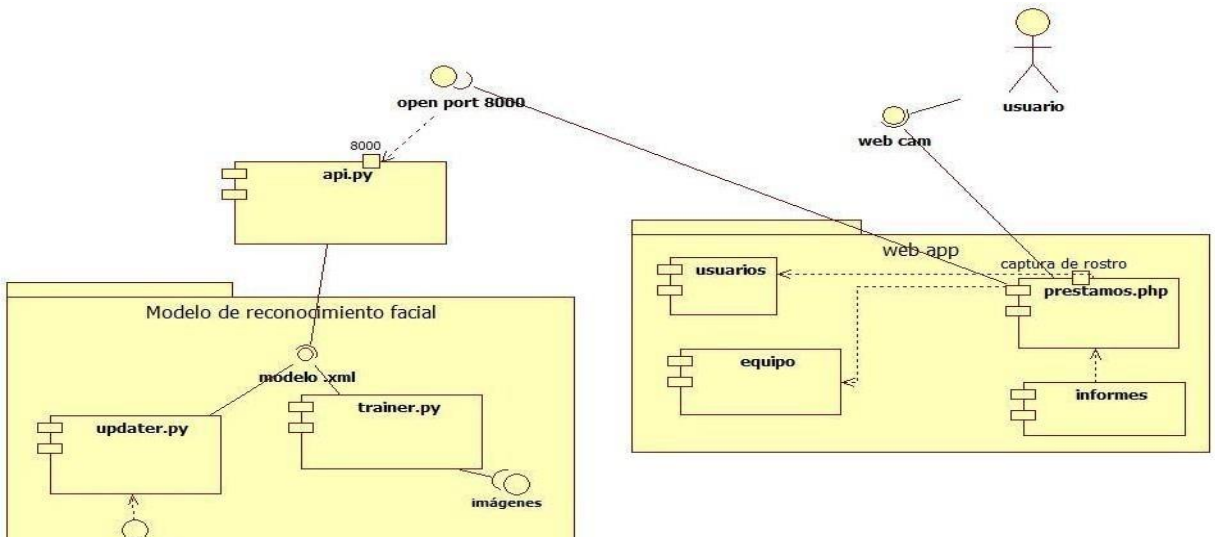

Figura 3. Diagrama de componentes.

h) Utilizar el modelo entrenado dentro de la API
Se lee el modelo guardado en la fase de entrenamiento
i) Extraer la imagen enviada a la API

En vista de que la imagen no es enviada como archivo, sino a través del método POST de HTTP, se utilizó una librería llamada *imageio* para leer la imagen que fue enviada. A continuación la imagen es convertida a escala de grises en un solo canal.

j) Buscar un rostro dentro de la imagen y predecir el resultado a través del modelo entrenado

Se busca un rostro dentro de la imagen empleando un clasificador de rostros frontales pre-entrenado de OpenCV. Una vez encontrado, se extraen los píxeles que conforman el rostro de la persona y se redimensiona a 150 píxeles de ancho por 150 píxeles de alto. Se envía la imagen redimensionada al modelo de reconocimiento facial entrenado, guardando el resultado obtenido.

k) Enviar el resultado como respuesta, si está dentro de un umbral aceptable.

El reconocedor de rostro nos devuelve dos datos: una etiqueta y una distancia. Si la distancia es menor a 65 devolvemos la etiqueta o el dato asociado a la etiqueta. La etiqueta es el número de matrícula o el número de empleado o nombre de la persona.

Implementación dentro de la aplicación web

l) Acceder a la cámara del equipo

Se añade a la aplicación web un espacio que será ocupado con el video que capture la cámara.

m) Tomar una fotografía cuando el usuario lo indique.

Se crea un botón que el usuario pulsará cuando esté listo para capturar su rostro.

n) Tratamiento de la imagen para poder ser enviada a través de protocolo HTTP

La imagen se convierte a escala de grises y se incluye en el formulario que se envía a través de HTTP.

o) Envío de la imagen en forma asíncrona

Se utiliza AJAX y se envía el formulario que incluye la imagen a la dirección de servidor que utiliza la API.
p) Recepción y tratamiento de la respuesta

Se recibe la respuesta (número de matrícula o número de empleado) y se utiliza para buscar el resto de los datos de la persona que está solicitando el equipo.

Resultados

Se elaboraron los módulos para crear el modelo de entrenamiento facial y su intercomunicación con la aplicación como se ve en la figura 3.

Se obtuvieron imágenes de: estudiantes del instituto, empleados del instituto, y se añadieron otras imágenes para evaluar la precisión del modelo.

No todos los estudiantes a los que se solicitaron sus fotografías las enviaron. Quienes no enviaron sus imágenes no mencionaron los motivos.

Los estudiantes que sí enviaron sus fotografías no siempre aportaron imágenes claras o útiles para el entrenamiento. Además las enviaron en diferentes formatos. Se tuvo que crear módulos para redimensionar y renombrar los archivos a fin de facilitar el preprocesamiento de las imágenes. En la figura 4 se puede observar un conjunto de imágenes proporcionadas por uno de los usuarios.

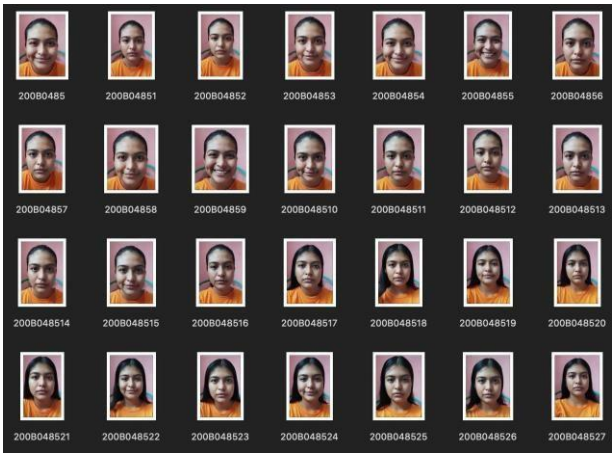

Figura 4. Imágenes aportadas por los usuarios para el entrenamiento.

Se obtuvieron los siguientes resultados en la fase de entrenamiento. Se consideran las métricas de *Precision, Recall, F1-Score* y *Accuracy* (Ver Tabla 2).

Tabla 2. Resultados obtenidos en el entrenamiento.

Persona	*Precision*	*Recall*	*F1-Score*
180B0106	1.00	1.00	1.00
200B0570	1.00	1.00	1.00
180B0108	0.60	1.00	0.75
200B0565	0.85	1.00	0.92
180B0115	0.92	1.00	0.96
180B0112	1.00	1.00	1.00
180B0107	1.00	0.50	0.67
200B0485	0.81	1.00	0.90
180B0724	1.00	1.00	1.00
200B0129	1.00	1.00	1.00
natividad	0.82	0.75	0.78
aurelio	1.00	1.00	1.00
wendy	1.00	0.75	0.86
gibran	0.75	1.00	0.86
isai	1.00	1.00	1.00
alex	0.86	0.75	0.80
Media ponderada	0.92	0.91	0.90
Accuracy			0.91

Para obtener los resultados anteriores se hizo uso de la biblioteca *sklearn.metric*s para Python. Además se corroboraron los resultados obtenidos, realizando el cálculo en forma manual, utilizando una Matriz de Confusión.

Para mejorar las métricas se hicieron diferentes pruebas variando el tamaño y la cantidad de los archivos enviados.

Se procesaron 399 fotografías y el modelo obtenido se guardó en un archivo XML ocupando 46 MB de espacio de almacenamiento.

La interfaz gráfica de la aplicación fue cambiada para que incluyera un espacio donde se mostrará la imagen que toma la cámara de video y se incluyó un botón para que el usuario (o un encargado) lo pulse cuando esté listo para hacer el reconocimiento. También se incluye un espacio para mostrar la imagen obtenida ya convertida a escala de grises. (Ver figura 5)

Figura 5. Interfaz gráfica obteniendo imagen de la cámara.

Se utilizó la función *time.time()* de Python para calcular la cantidad de milisegundos que la API se demora en hacer el reconocimiento facial. Se comienza a contar el tiempo desde el momento en que la función de reconocimiento es invocada hasta el momento en que la imagen es reconocida. Para cada llamada se obtiene la clave de etiqueta obtenida, la etiqueta (el nombre o matrícula del usuario) y el tiempo empleado. En la figura 6 se observa la salida en consola de 3 llamadas a la API. Como se puede observar en cada caso el reconocimiento se consiguió en menos de 100 milisegundos. Las pruebas se hicieron en una computadora con procesador Intel Core i5 a 2.7 GHz.

```
--------------------------
127.0.0.1 - - [06/Sep/2021 17:14:37] "POST /recognizer HTTP/1.1" 200 23
--------------------------
Clave de etiqueta obtenida: 10
Etiqueta obtenida: gibran
Tiempo de reconocimiento: 55.088043212890625 ms.
--------------------------
127.0.0.1 - - [06/Sep/2021 17:15:32] "POST /recognizer HTTP/1.1" 200 23
--------------------------
Clave de etiqueta obtenida: 10
Etiqueta obtenida: gibran
Tiempo de reconocimiento: 49.53885078430176 ms.
--------------------------
127.0.0.1 - - [06/Sep/2021 17:15:35] "POST /recognizer HTTP/1.1" 200 23
--------------------------
Clave de etiqueta obtenida: 10
Etiqueta obtenida: gibran
Tiempo de reconocimiento: 63.043832778930664 ms.
--------------------------
```

Figura 6. Tiempo de reconocimiento

DISCUSIÓN Y ANÁLISIS DE RESULTADOS

Los datos obtenidos en la evaluación del modelo nos permiten ver que está listo para su uso con resultados aceptables. La media ponderada obtenida para la métrica

precision fue de 0.92, para *recall* fue de 0.91 y para *f1* fue de 0.90. Estos resultados son aceptables. El método usado para obtenerlos nos permite observar los valores en forma independiente, lo que hace posible mejorar la precisión del modelo modificando las imágenes de entrada de esa persona. Se deberá comparar los datos arrojados por la evaluación del modelo con las observaciones que se hagan durante el uso normal del sistema.

El cambio realizado a la interfaz gráfica facilita y agiliza la interacción entre el usuario y el sistema.

El tiempo de reconocimiento es muy corto, lo que permitirá un uso fluido de la aplicación.

El uso de esta herramienta de reconocimiento facial en el sistema de préstamo de equipo permitirá agilizar ese proceso del departamento de Desarrollo Académico, con un considerable ahorro económico al no depender de ninguna tecnología de pago.

En contraste con los autores [16], ellos alcanzan un rango de predicción del rostro entre el 70 y el 90%, sin embargo, se muestran muy conscientes que no se tiene un nivel de confianza del 100%, lo cual puede marcar como resultado una identificación errónea de la persona.

Se puede apreciar que una predicción mayor al 80% es un parámetro aceptable en términos generales, aunque sin lugar a dudas que el nivel ideal es el cercano al 100%. Por lo tanto, el modelo de entrenamiento de reconocimiento facial con una precisión tan alta como el 90% se considera factible.

CONCLUSIONES

LBPH puede ser usado en aplicaciones que involucren reconocimiento facial con resultados muy aceptables. El uso de este algoritmo demostró que es posible agregar ágilmente nuevos rostros a un modelo ya entrenado. La evaluación del modelo entrenado contribuyó a determinar si el modelo está listo para su uso por parte de un software de aplicación e incluso nos permitió observar al usuario cuyas imágenes pudieran estar afectando al desempeño del modelo.

El uso de HUG para el desarrollo de la API de reconocimiento facial fue aceptable. La implementación fue simple y su desempeño fue estable todo el tiempo.

Dentro de la aplicación web es necesario realizar preprocesamiento de la imagen tomada a fin de enviarla correctamente al servidor HUG.

La separación del sistema en capas y su comunicación a través de una API fue una elección correcta y hace posible su uso por parte de otras tecnologías como las móviles.

Con el fin de mejorar aún más la seguridad del sistema, se estudia la incorporación de detectores de presencia por infrarrojos, y para mejorar más la precisión se analiza el manejo de un módulo de corrección del posicionamiento del rostro.

AGRADECIMIENTOS

Se agradece especialmente al departamento de Desarrollo Académico del campus Acayucan del TecNM las facilidades otorgadas para realizar éste proyecto.

BIBLIOGRAFÍA

[1] Colmenarez, A. J. & Huang, T. S. (1998). Face detection and recognition. In Face Recognition (pp. 174-185). Springer, Berlin, Heidelberg.

[2] Raza, M., Iqbal, M., Sharif, M., & Haider, W. (2012). A survey of password attacks and comparative analysis on methods for secure authentication. World Applied Sciences Journal, 19(4), 439-444.

[3] Kumar, S. & Walia, E. (2011). Analysis of various biometric techniques. International Journal of Computer Science and Information Technologies, 2(4), 1595-1597.

[4] Masupha, L., Zuva, T., Ngwira, S., & Esan, O. (2015). Face recognition techniques, their advantages, disadvantages and performance evaluation. In 2015 International Conference on Computing, Communication and Security (ICCCS) (pp. 1-5). IEEE.

[5] Kumar, A., Kaur, A., & Kumar, M. (2019). Face detection techniques: a review. Artificial Intelligence Review, 52(2), 927-948.

[6] Tistarelli, M. & Grosso, E. (2010). Human Face Analysis: From Identity to Emotion and Intention Recognition. In International Conference on Ethics and Policy of Biometrics (pp. 76-88). Springer, Berlin, Heidelberg.

[7] Zhang, B., Gao, Y., Zhao, S., & Liu, J. (2009). Local derivative pattern versus local binary pattern: face

recognition with high-order local pattern descriptor. IEEE transactions on image processing, 19(2), 533-544.

[8] Shetty, A. B., & Rebeiro, J. (2021). Facial Recognition using Haar Cascade and LBP Classifiers. Global Transitions Proceedings.

[9] Deeba, F., Ahmed, A., Memon, H., Dharejo, F. A., & Ghaffar, A. (2019). LBPH-based enhanced real-time face recognition. Int J Adv Comput Sci Appl, 10(5), 274-280.

[10] Zhang, C., & Zhang, Z. (2010). A survey of recent advances in face detection.

[11] Aguillón, M. (2020). Sistema para la detección de dolor por medio de expresiones faciales. [Tesis de Maestría]. CICESE, Baja California, México. Recuperado de: https://cicese.repositorioinstitucional.mx/jspui/bitstream/1007/3222/1/Tesis_Mar%C3%ADa%20Lourdes_Aguillon%20Ruiz_13_feb_2020.pdf

[12] Masek, P., & Thulin, M. (2015). Evaluation of face recognition apis and libraries.

[13] Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. University of California, Irvine.

[14] Oliphant, T. E. (2007). Python for scientific computing. Computing in science & engineering, 9(3), 10-20.

[15] Powers, D. M. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061.

[16]Llapapasca, M.M. Creación de una librería de software de reconocimiento facial enfocado a la identificación de trabajadores de una empresa. Universidad Tecnológica de Lima Perú

ROLES

ROL DE CONTRIBUCIÓN	AUTOR
Conceptualización y Software	Francisco Gibrán García Candelario
Recursos	Natividad Juárez González
Redacción (Revisión y edición)	José Aurelio Ramírez González
Supervisión	Wendy Carranza Díaz