UNIVERSIDAD CATÓLICA BOLIVIANA "SAN PABLO"
UNIDAD ACADÉMICA REGIONAL TARIJA

PROYECTO FINAL DE ASIGNATURA

"DETECTOR DE EMOCIONES MEDIANTE VISIÓN POR COMPUTADORA"

ESTUDIANTES: Diego Fernando Perales Salinas
Andrés La Madrid Salinas
Dagiro Wilfredo Fernandez Giron
Jorge Luis Estrada Guerra
Pablo Fernando Aban Palacios
Paolo Alejandro Velarde Ramírez

ASIGNATURA: SIS-341 Sistemas Inteligentes [Par.1]

DOCENTE: Ing. Helmer Fellman Mendoza Jurado

Tarija-Bolivia

Introducción.

El reconocimiento de emociones mediante el rostro es una tecnología que ha experimentado un importante desarrollo en los últimos años. Gracias a los avances en la visión por computadora, es posible identificar las emociones básicas de una persona a partir de su expresión facial. Esta tecnología tiene un gran potencial en una variedad de aplicaciones, como la atención al cliente, la educación, la salud mental y la seguridad.

En los últimos años, el reconocimiento de emociones mediante el rostro ha experimentado un gran progreso gracias al desarrollo de las redes neuronales profundas más en concreto de las redes neuronales convulsiónales. Las redes neuronales convulsiónales están diseñadas específicamente para trabajar con datos de tipo gráfico, como imágenes y han demostrado ser muy eficaces en tareas relacionadas con la visión por computadora, lo que las hace muy adecuadas para tareas de reconocimiento de patrones como el reconocimiento de emociones.

Este proyecto tiene como objetivo desarrollar un sistema de reconocimiento de emociones mediante el rostro basado en redes neuronales convulsiónales.

El sistema o modelo se basará principalmente en tres etapas:

- Capturar Emociones. El sistema capturara imágenes de rostros en tiempo real desde la cámara y las almacenara en carpetas asociadas a una emoción en particular, con el objetivo de construir un conjunto de datos para el reconocimiento de emociones.
- Entrenamiento de Emociones. Una vez almacenadas las imágenes para cada emoción, el sistema cargará las imágenes de los rostros etiquetadas con diferentes emociones y entrenará modelos de reconocimiento facial utilizando tres métodos diferentes (EigenFaces, FisherFaces y LBPH), y almacenará los modelos entrenados en archivos XML.
- Reconocimiento de Emociones. Una vez terminado el entrenamiento, el sistema realizará el reconocimiento de emociones en tiempo real a través de la cámara utilizando cualquiera de los tres métodos mencionados anteriormente, evaluando el rendimiento del modelo utilizando distintas métricas.

Trabajos relacionados.

Algunos de los primeros trabajos en el campo de la detección de emociones faciales utilizando IA se centraron en el uso de técnicas de reconocimiento de patrones tradicionales, como eigenfaces y fisherfaces. Estas técnicas se basan en la idea de que las caras pueden ser representadas como una

combinación lineal de características faciales básicas. Sin embargo, estas técnicas tienen algunas limitaciones, como la sensibilidad a las variaciones en la expresión facial y la iluminación.

En los últimos años, se ha producido un aumento de los trabajos que utilizan técnicas de ML (Machine Learning) y DL (Deep Learning) para la detección de emociones faciales. Estas técnicas tienen la ventaja de ser más robustas a las variaciones en las condiciones de iluminación y expresión facial.

A continuación, se presentan algunos ejemplos específicos de trabajos relacionados y avances en la aplicación de inteligencia artificial, modelos de machine learning, deep learning y sistemas inteligentes para problemas similares en el contexto de visión por computadora y detección de emociones:

- Google solo ve códigos. Por lo tanto, necesitan aprender cuáles son las características de la fotografía de un perro para comprender cuando están allí. Ahí es donde entra la visión computacional. Esta tecnología te permite entrenar a tu computadora para que reconozca patrones de colores y formas en las imágenes. De esta manera, las máquinas están más cerca de la visión humana y pueden tomar decisiones según lo que ven. Por lo tanto, la aplicación no solo reconoce las fotos de perros, sino que también reconoce las fotos de tu perro. No solo reconoce fotos de personas en general, sino que también reconoce fotos de tu familia o amigos. Y cuanto más le digan los usuarios a los robots quién o qué aparece en las imágenes, más aprenden. De esta forma, Google Photos puede organizar y agrupar las fotos que guardas, para que puedas encontrarlas con una simple búsqueda.
- Otro claro ejemplo que se puede apreciar en la actualidad sobre la visión por computadora es la plataforma Pinterest, el enfoque de la plataforma es lo visual. Por lo tanto, la visión computacional es la principal tecnología de Inteligencia Artificial usada para mejorar la experiencia del usuario. Pinterest Lens, por ejemplo —que te permite usar la cámara de tu celular en búsquedas— ve imágenes casi como una persona. Los robots necesitan identificar patrones en las imágenes para hacer recomendaciones alineadas con la investigación y sus gustos e intereses.

Avances.

Algunos de los avances más importantes en el campo de la detección de emociones faciales utilizando IA incluyen:

- El desarrollo de nuevos modelos de ML y DL que son más precisos y robustos.

- El aumento del tamaño de los conjuntos de datos de entrenamiento, lo que permite a los modelos aprender patrones más complejos.
- La mejora de los algoritmos de pre procesamiento de imágenes, que ayudan a eliminar el ruido y las variaciones en las condiciones de iluminación.

Estos avances han llevado al desarrollo de sistemas de detección de emociones faciales que se utilizan en una variedad de aplicaciones, como la atención médica, la seguridad y el entretenimiento.

Aplicaciones.

La detección de emociones faciales tiene una variedad de aplicaciones potenciales, entre las que se incluyen:

- Atención médica: La detección de emociones faciales podría utilizarse para diagnosticar trastornos psicológicos, evaluar el estado de ánimo de los pacientes y personalizar la atención médica.
- Seguridad: La detección de emociones faciales podría utilizarse para detectar personas que se sienten ansiosas, enojadas o hostiles, lo que podría ayudar a prevenir la violencia.
- Entretenimiento: La detección de emociones faciales podría utilizarse para crear juegos y aplicaciones que se adapten al estado de ánimo del usuario.

Se espera que la aplicación de IA, ML, DL y sistemas inteligentes para problemas similares en el contexto de visión por computadora y detección de emociones continúe creciendo en los próximos años. Estos avances conducirán al desarrollo de sistemas de detección de emociones faciales más precisos y robustos, que se utilizarán en una gama aún más amplia de aplicaciones.

Metodologías y técnicas más relevantes.

El sistema sigue una metodología de aprendizaje automático supervisado. En este tipo de aprendizaje, el sistema se entrena con un conjunto de datos de entrenamiento que contiene datos etiquetados. En este caso, el conjunto de datos de entrenamiento contiene imágenes de rostros humanos con etiquetas de emociones.

El sistema se divide en tres partes, la primera parte del código se encarga de recopilar las imágenes de rostros para cada emoción. Para ello, utiliza el algoritmo HaarCascade para detectar rostros en imágenes capturadas desde una cámara web. Una vez que el algoritmo HaarCascade detecta un rostro, el código recorta y redimensiona el área del rostro para ser almacenada como imagen. Las imágenes de los rostros se guardan en una carpeta con el nombre de la emoción correspondiente.

La segunda parte se encarga de entrenar un modelo de reconocimiento facial para cada emoción. Para ello, utiliza tres métodos de reconocimiento facial diferentes:

- EigenFaces: Este método se basa en la descomposición de los datos en autovectores.
- FisherFaces: Este método se basa en la selección de los autovectores que capturan la mayor variabilidad entre las clases.
- LBPH: Este método se basa en la utilización de características de textura locales.

Cada método tiene sus propias ventajas y desventajas. EigenFaces es un método simple y eficiente, pero puede ser menos preciso que otros métodos. FisherFaces es más preciso que EigenFaces, pero puede ser más complejo y requerir más datos de entrenamiento. LBPH es un método robusto a las variaciones en la iluminación y la orientación del rostro, pero puede ser menos preciso que otros métodos.

La tercera parte se encarga de probar el modelo de reconocimiento facial. Para ello, el código utiliza una webcam para capturar imágenes de rostros. El modelo de reconocimiento facial se utiliza para predecir la emoción del rostro capturado.

Diseño del Sistema.

Arquitectura General.

El sistema se compone de los siguientes módulos:

1. Recolector de datos: Este módulo se encarga de recopilar las imágenes de rostros para cada emoción. Para ello, utiliza el algoritmo HaarCascade para detectar rostros en imágenes capturadas desde una cámara web. Una vez que el algoritmo HaarCascade detecta un rostro, el recolector de datos recorta y redimensiona el área del rostro para ser almacenada como imagen. Las imágenes de los rostros se guardan en una carpeta con el nombre de la emoción correspondiente.
2. Entrenador de modelos: Este módulo se encarga de entrenar un modelo de reconocimiento facial para cada emoción. Para ello, utiliza tres métodos de reconocimiento facial diferentes:
    - EigenFaces: Este método se basa en la descomposición de los datos en autovectores.
    - FisherFaces: Este método se basa en la selección de los autovectores que capturan la mayor variabilidad entre las clases.
    - LBPH: Este método se basa en la utilización de características de textura locales.

3. Probador de modelos: Este módulo se encarga de probar el modelo de reconocimiento facial. Para ello, el código utiliza una webcam para capturar imágenes de rostros. El modelo de reconocimiento facial se utiliza para predecir la emoción del rostro capturado.

Explicación de las Decisiones de Diseño.

Las decisiones de diseño que se han tomado en el desarrollo de este sistema inteligente se basan en los siguientes criterios:

- Sencillez: El sistema se diseñó para ser lo más sencillo posible de entender y de implementar.
- Eficiencia: El sistema se diseñó para ser eficiente en cuanto a recursos de hardware y software.
- Precisión: El sistema se ha diseñado para ser lo más preciso posible en la predicción de las emociones.

En base a estos criterios, se han tomado las siguientes decisiones de diseño:

- El algoritmo HaarCascade se ha utilizado para la detección de rostros debido a su sencillez y eficiencia.
- Se utilizaron tres métodos de reconocimiento facial diferentes para evaluar el rendimiento del sistema
- El conjunto de datos de entrenamiento se ha recopilado utilizando una cámara web para garantizar que el sistema sea robusto a las variaciones e la iluminación y la orientación del rostro.

Implementación.

Proceso de Implementación:

1. **Adquisición y Preparación de Datos.**
   Se capturan imágenes de rostros asociados a diferentes emociones utilizando una cámara en tiempo real, posteriormente cada imagen se etiqueta con la emoción correspondiente (Felicidad, Enojo, Sorpresa y Tristeza).
2. **Pre procesamiento de Datos.**
   Las imágenes se re dimensionan para tener un tamaño consistente y luego las imágenes se convierten a escala de grises para reducir la complejidad y facilitar el procesamiento.
3. **Entrenamiento del Modelo.**

El conjunto de datos se divide en conjuntos de entrenamiento y prueba luego el modelo de reconocimiento facial se entrena utilizando el conjunto de entrenamiento.

Se entrena por los tres métodos mas populares para el reconocimiento facial:

- EigenFaces. Funciona representando las caras como una combinación lineal de un pequeño número de características faciales básicas. Se utiliza para reducir la dimensionalidad de las imágenes faciales y capturar las características más distintivas. Utiliza la descomposición en valores singulares (SVD) para encontrar los auto vectores de la matriz de covarianza de las imágenes faciales, en términos más simples, se usan matemáticas para encontrar patrones importantes en las caras, llamados “eigenfaces” y luego se representa cada cara como una combinación de estos patrones. Esto hace que sea más fácil entender y comprara diferentes rostros para el reconocimiento facial. EigenFaces es eficaz, pero puede ser sensible a las variaciones en la iluminación y la pose.
- FisherFaces: Se basa en la idea de que las caras pueden ser representadas como una combinación lineal de características faciales básicas. Estas características faciales son las que permiten distinguir entre diferentes personas, en este caso entre las emociones, independientemente de la iluminación o el ángulo de visión. El programa busca características específicas en los rostros, como la posición de los ojos, nariz y boca, para entender lo que hace único a cada rostro o emoción. Fisherfaces aprende a distinguir caras al buscar patrones específicos, por ejemplo, en el caso de las emociones, busca la sonrisa, el gesto en los ojos o nariz. Este método tiende a funcionar mejor que EigenFaces cuando hay variaciones en la iluminación y la pose.
- LBPH (Local Binary Pattern Histograms): es un enfoque basado en patrones locales binarios que captura información sobre la textura de la imagen. Se utiliza para reconocimiento facial en situaciones donde la iluminación puede variar significativamente. LBPH opera a nivel de píxeles, describiendo la textura local de la imagen mediante la comparación de los valores de intensidad de los píxeles con el valor del píxel central. Luego, se genera un patrón binario que se utiliza para construir un histograma local. Estos histogramas se utilizan para representar las imágenes faciales. LBPH es robusto a variaciones en la iluminación y es menos sensible a cambios en la pose en comparación con EigenFaces y FisherFaces.

4. **Almacenado del Modelo Entrenado**.

   El modelo entrenado se almacena en un archivo XML para cada emoción.

5. **Detección en Tiempo Real.**

   Se inicia la captura del fotograma desde la cámara en tiempo real, se utiliza un clasificador en casada (HearCascade) para detectar rostros en cada fotograma luego se escoge y aplica el modelo entrenado (EigenFaces o FisherFaces o LBPH) para predecir la emoción en cada rostro detectado.

   Para decidir si una cara es reconocida o no utilizamos umbrales fijos, estos umbrales actúan como un criterio de aceptación. Si la medida de similitud entre el rostro reconocido y el rostro en el conjunto de entrenamiento supera el umbral, se toman ciertas acciones (como etiquetar el rostro como "Neutral o Compuesto" en este caso). Si la medida de similitud es inferior al umbral, se toman otras acciones (como mostrar el nombre de la emoción y dibujar un rectángulo en el fotograma).

6. **Visualización y Evaluación en Tiempo Real.**

   Se dibujan rectángulos alrededor de los rostros detectados y se muestra la emoción predicha. Una vez concluida las predicciones de las emociones se procede a recopilar métricas de evaluación como matriz de confusión sensibilidad, precisión y F1-score.

7. **Análisis de Resultados.**

   Se analizan las métricas para evaluar el rendimiento del modelo en la tarea de reconocimiento de emociones y se muestran en tiempo real la matriz de confusión y se calcula el promedio de los F1-score para poder tener una idea de la calidad del modelo.

8. **Iteración y Mejora.**

   Si es necesario, se ajustan los parámetros del modelo o se consideran mejoras para optimizar el rendimiento del sistema.

9. **Documentación y Presentación.**

   El código está documentado para explicar las funciones y la lógica que se sigue y también se presentan los resultados, métricas y visualización de manera clara y comprensibles.

Código Fuente.

Primera Parte – capturar_emociones,

```python
import cv2
import os
import numpy as np
import time

# Función para obtener y almacenar el modelo de reconocimiento facial según el método especificado.
def obtenerModelo(method,facesData,labels):
    if method == 'EigenFaces': emotion_recognizer = cv2.face.EigenFaceRecognizer_create()
    if method == 'FisherFaces': emotion_recognizer = cv2.face.FisherFaceRecognizer_create()
    if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()

    # Entrenando el reconocedor de rostros
    print("Entrenando ( "+method+" )...")
    inicio = time.time()
    emotion_recognizer.train(facesData, np.array(labels))
    tiempoEntrenamiento = time.time()-inicio
    print("Tiempo de entrenamiento ( "+method+" ): ", tiempoEntrenamiento)

    # Almacenando el modelo obtenido
    emotion_recognizer.write("modelo"+method+".xml")

dataPath = 'C:/Users/HP/Desktop/Inteligentes/Reconocimiento Emociones/Data' # Ruta donde esta almacenado Data
emotionList = os.listdir(dataPath)
print('Lista de personas: ', emotionList)

# Lista de emociones (nombres de carpetas)
labels = []     # Lista para almacenar las etiquetas asociadas a cada rostro
facesData = []     # Lista para almacenar las imágenes de rostros
label = 0

# Iterar sobre las carpetas de emociones
for nameDir in emotionList:
    emotionPath = dataPath + '/' + nameDir
    print('Leyendo las imágenes')

    # Iterar sobre las imágenes de rostros en la carpeta de la emoción actual
    for fileName in os.listdir(emotionPath):
        labels.append(label)
        facesData.append(cv2.imread(emotionPath+'/'+fileName,0))
        image = cv2.imread(emotionPath+'/'+fileName,0)

    label = label + 1

# Obtener y almacenar el modelo para cada método de reconocimiento facial
obtenerModelo('EigenFaces',facesData,labels)
obtenerModelo('FisherFaces',facesData,labels)
obtenerModelo('LBPH',facesData,labels)

#------AQUI UNA VEZ GUARDADOS LOS ROSTROS, SE EJECUTA ESTO PARA QUE EL MODELO DE ENTRENE
```

Segunda Parte – entrenamiento_emociones.

Tercera Parte – reconocimiento_emociones.

capturar_emociones.py | entrenamiento_emociones.py | reconocimiento_emociones.py

```python
import cv2        # OpenCV para procesamiento de imágenes y visión por computadora
import os         # Módulo para interactuar con el sistema operativo
import numpy as np

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def emotionImage(emotion):
    # Asignar la imagen del emoji correspondiente a la emoción proporcionada
    if emotion == 'Felicidad': image = cv2.imread('Emojis/felicidad.jpeg')
    if emotion == 'Enojo': image = cv2.imread('Emojis/enojo.jpeg')
    if emotion == 'Sorpresa': image = cv2.imread('Emojis/sorpresa.jpeg')
    if emotion == 'Tristeza': image = cv2.imread('Emojis/tristeza.jpeg')
    return image

dataPath = 'C:/Users/HP/Desktop/Inteligentes/Proyecto_Reconocimiento_Emociones/Data'
imagePaths = os.listdir(dataPath)
print('imagePaths=', imagePaths)


# ---------- Métodos usados para el entrenamiento y lectura del modelo ----------
#method = 'EigenFaces'     # ESTE METODO NO DA TAN BUENOS RESULTADOS AL RECONOCER LOS ROSTROS
#method = 'FisherFaces'    # ESTE MEJORA UN POCO
method = 'LBPH'            # DE LOS TRES ESTE ES EL MAS OPTIMO

if method == 'EigenFaces':
    emotion_recognizer = cv2.face.EigenFaceRecognizer_create()
elif method == 'FisherFaces':
    emotion_recognizer = cv2.face.FisherFaceRecognizer_create()
elif method == 'LBPH':
    emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()

# Leyendo el modelo
# Cargar el modelo previamente entrenado para el método seleccionado
emotion_recognizer.read(f"modelo{method}.xml")

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
# Crear un clasificador en cascada para la detección de rostros
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Inicializamos las variables
etiquetas_reales = []
etiquetas_predichas = []

# Lista de emociones
emociones = ['Felicidad', 'Enojo', 'Tristeza', 'Sorpresa']

# Inicializar matrices de confusión para cada emoción
matrices_confusion = {emo: np.zeros((len(emociones), len(emociones)), dtype=int) for emo in emociones}


# Bucle principal para el procesamiento de imágenes continuo
while True:
    ret, frame = cap.read()
    if ret == False:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)     # Convertir la imagen a escala de grises para el procesamiento facial
    # Realizar una copia de la imagen en escala de grises para su posterior visualización
    auxFrame = gray.copy()

    nFrame = cv2.hconcat([frame, np.zeros((480,300,3),dtype=np.uint8)])

    faces = faceClassif.detectMultiScale(gray, 1.3, 5)
```

```python
    # Comprueba el método utilizado para el reconocimiento de emociones
    for (x, y, w, h) in faces:
        rostro = auxFrame[y:y+h, x:x+w]
        rostro = cv2.resize(rostro, (150, 150), interpolation=cv2.INTER_CUBIC)
        result = emotion_recognizer.predict(rostro)

        # Inicializamos la variable
        label = 0

        # Asumiendo que 'label' es la emoción real
        etiquetas_reales.append(label)
        etiquetas_predichas.append(result[0])

        # Actualizar matriz de confusión correspondiente a la emoción
        matrices_confusion[emociones[label]][result[0]] += 1


        cv2.putText(frame, '{}'.format(result), (x, y-5), 1, 1.3, (255, 255, 0), 1, cv2.LINE_AA)


        if method == 'EigenFaces':
            # Verifica si la similitud entre el rostro reconocido y el rostro de entrenamiento es menor que un umbral (5700)
            # El umbral actúa como un criterio de aceptación para el rostro reconocido, en este caso con EigenFaces trabaja con umbrales arriba de 1000
            if result[1] < 5700:
                # Muestra el nombre de la emoción sobre el fotograma
                cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x, y-25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                image = emotionImage(imagePaths[result[0]])
                nFrame = cv2.hconcat([frame, image])
            else:
                cv2.putText(frame, 'Neutral o Compuesto', (x, y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
                nFrame = cv2.hconcat([frame, np.zeros((480,300,3),dtype=np.uint8)])


        elif method == 'FisherFaces':
            # Verifica si la similitud entre el rostro reconocido y el rostro de entrenamiento es menor que un umbral (500)
            # El umbral actúa como un criterio de aceptación para el rostro reconocido, en este caso con FisherFaces trabaja con umbrales arriba de 100
            if result[1] < 500:
                # Muestra el nombre de la emoción sobre el fotograma
                cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x, y-25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                image = emotionImage(imagePaths[result[0]])
                nFrame = cv2.hconcat([frame, image])
            else:
                cv2.putText(frame, 'Neutral o Compuesto', (x, y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
                nFrame = cv2.hconcat([frame, np.zeros((480,300,3),dtype=np.uint8)])



        elif method == 'LBPH':
            # Verifica si la similitud entre el rostro reconocido y el rostro de entrenamiento es menor que un umbral (70)
            # El umbral actúa como un criterio de aceptación para el rostro reconocido, en este caso con LBPH trabaja con umbrales arriba de 10
            if result[1] < 70:
                # Muestra el nombre de la emoción sobre el fotograma
                cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x, y-25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                image = emotionImage(imagePaths[result[0]])
                nFrame = cv2.hconcat([frame, image])
            else:
                cv2.putText(frame, 'Neutral o Compuesto', (x, y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
                nFrame = cv2.hconcat([frame, np.zeros((480,300,3),dtype=np.uint8)])
```

```
    # Espera la pulsación de la tecla ESC (27) o hasta que se capturen 200 rostros
    cv2.imshow('nFrame', nFrame)
    k = cv2.waitKey(1)
    if k == 27:
        break

# Cierra la ventana de la camara
cap.release()
cv2.destroyAllWindows()


# Calcular métricas para cada emocion

# Calcula una matriz de confucion para tener una aproximacion de que emocion se registra mas
# Ejemplo
    #   [[139 139 139 139]   [[enojo     139      139        139]
    #   [ 61  61  61  61]    [ 61     felicidad   61          61]
    #   [ 45  45  45  45]    [ 45         45   sorpresa       45]
    #   [  4   4   4   4]]   [  4         4        4    tristeza]]

for emociones, matriz_confusion in matrices_confusion.items():
    print(f'Matriz de Confusión:\n{matriz_confusion}')
    break


# Calcular Sensibilidad y el Soporte
for emocion, matriz_confusion in matrices_confusion.items():
    # Sensibilidad, indica la proporción de casos de cada emoción que el modelo ha identificado correctamente
    sensibilidad = matriz_confusion[0, 0] / np.sum(matriz_confusion[0, :])
    # Precisión, indica la proporción de casos identificados como Enojo que son realmente Enojo
    precision = matriz_confusion[0, 0] / np.sum(matriz_confusion[:, 0])
    # F1-score, combina la precisión y la sensibilidad en una sola métrica
    f1_score = 2 * (precision * sensibilidad) / (precision + sensibilidad)
    print('--------------------ENOJO-----------------------')
    print(f'Sensibilidad: {sensibilidad}')
    print(f'Precisión: {precision}')
    print(f'F1-score: {f1_score}')
    break

for emocion, matriz_confusion in matrices_confusion.items():
    # Sensibilidad, indica la proporción de casos de cada emoción que el modelo ha identificado correctamente
    sensibilidad = matriz_confusion[1, 1] / np.sum(matriz_confusion[0, :])
    # Precisión, indica la proporción de casos identificados como Felicidad que son realmente Felicidad
    precision = matriz_confusion[1, 1] / np.sum(matriz_confusion[:, 0])
    # F1-score, combina la precisión y la sensibilidad en una sola métrica
    f1_score = 2 * (precision * sensibilidad) / (precision + sensibilidad)
    print('--------------------FELICIDAD-----------------------')
    print(f'Sensibilidad: {sensibilidad}')
    print(f'Precisión: {precision}')
    print(f'F1-score: {f1_score}')
    break

for emocion, matriz_confusion in matrices_confusion.items():
    # Sensibilidad, indica la proporción de casos de cada emoción que el modelo ha identificado correctamente
    sensibilidad = matriz_confusion[2, 2] / np.sum(matriz_confusion[0, :])
    # Precisión, indica la proporción de casos identificados como Sorpresa que son realmente Sorpresa
    precision = matriz_confusion[2, 2] / np.sum(matriz_confusion[:, 0])
    # F1-score, combina la precisión y la sensibilidad en una sola métrica
    f1_score = 2 * (precision * sensibilidad) / (precision + sensibilidad)
    print('--------------------SORPRESA-----------------------')
    print(f'Sensibilidad: {sensibilidad}')
    print(f'Precisión: {precision}')
    print(f'F1-score: {f1_score}')
    break

for emocion, matriz_confusion in matrices_confusion.items():
    # Sensibilidad, indica la proporción de casos de cada emoción que el modelo ha identificado correctamente
    sensibilidad = matriz_confusion[3, 3] / np.sum(matriz_confusion[0, :])
    # Precisión, indica la proporción de casos identificados como Tristeza que son realmente Tristeza
    precision = matriz_confusion[3, 3] / np.sum(matriz_confusion[:, 0])
    # F1-score, combina la precisión y la sensibilidad en una sola métrica donde si se acerca a 1 es optimo y si esta cerca de 0 no lo es
    f1_score = 2 * (precision * sensibilidad) / (precision + sensibilidad)
    print('--------------------TRISTEZA-----------------------')
    print(f'Sensibilidad: {sensibilidad}')
    print(f'Precisión: {precision}')
    print(f'F1-score: {f1_score}')
    break
```

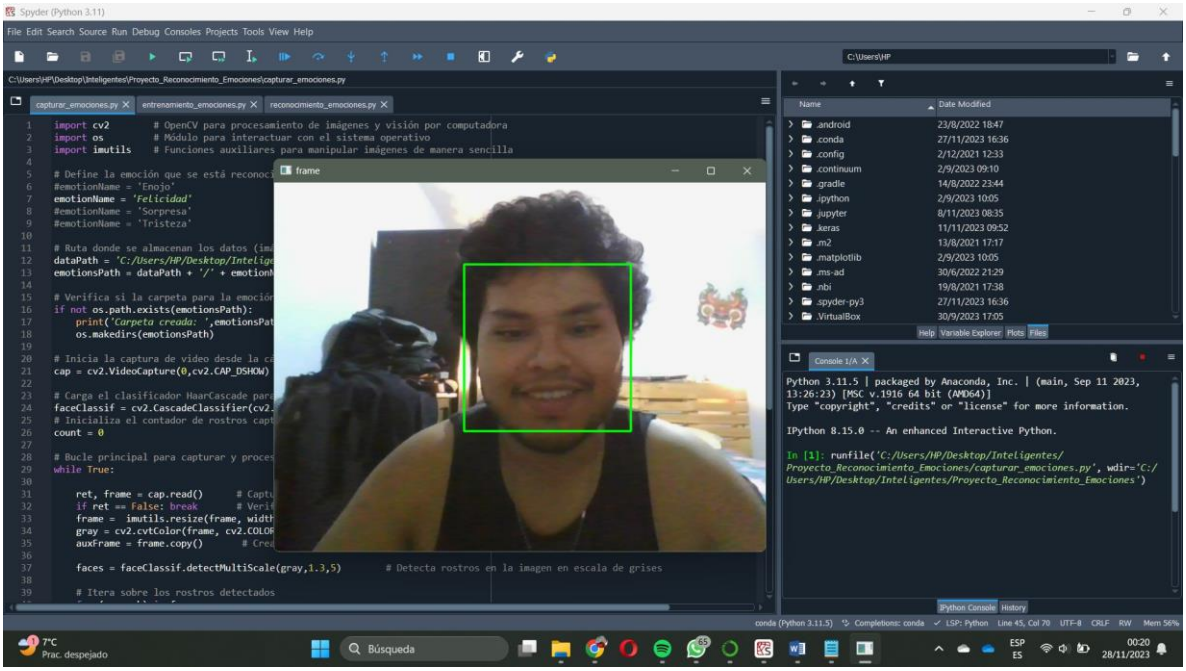
```
# Calcula el promedio de los scores F1
promedio_f1 = np.mean(f1_score)
print('----------------------------------------')
# Para tener una idea de que tan optimo es el modelo, podemos tomar el promedio de los f1_score donde si esta cerca de 0 es poco optimo y si esta
# cerca a 1 si lo es
# aunque dependera mucho del tiempo que se ejecute el modelo
print(f'Promedio total de F1-score: {promedio_f1}')

 # TODAS LAS METRICAS CALCULADAS DEPENDERAN MUCHO DEL TIEMPO QUE SE REGISTRE CADA EMOCION


#---------------AQUI SE SUBEN NUEVOS ROSTROS PARA VER SI EXISTE SIMILITUD CON LOS QUE ENTRENAMOS
```

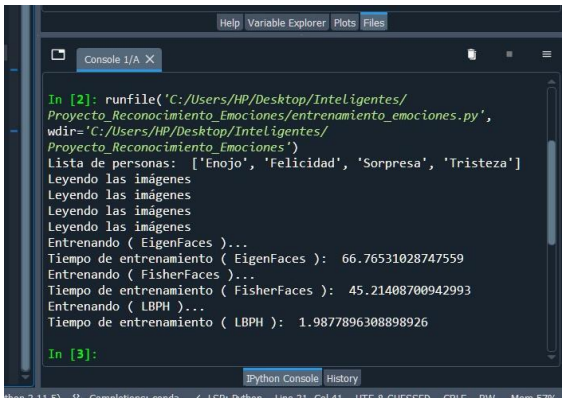
Ejemplo de Entrada.

El sistema saca fotografías de acuerdo a la emoción que necesite



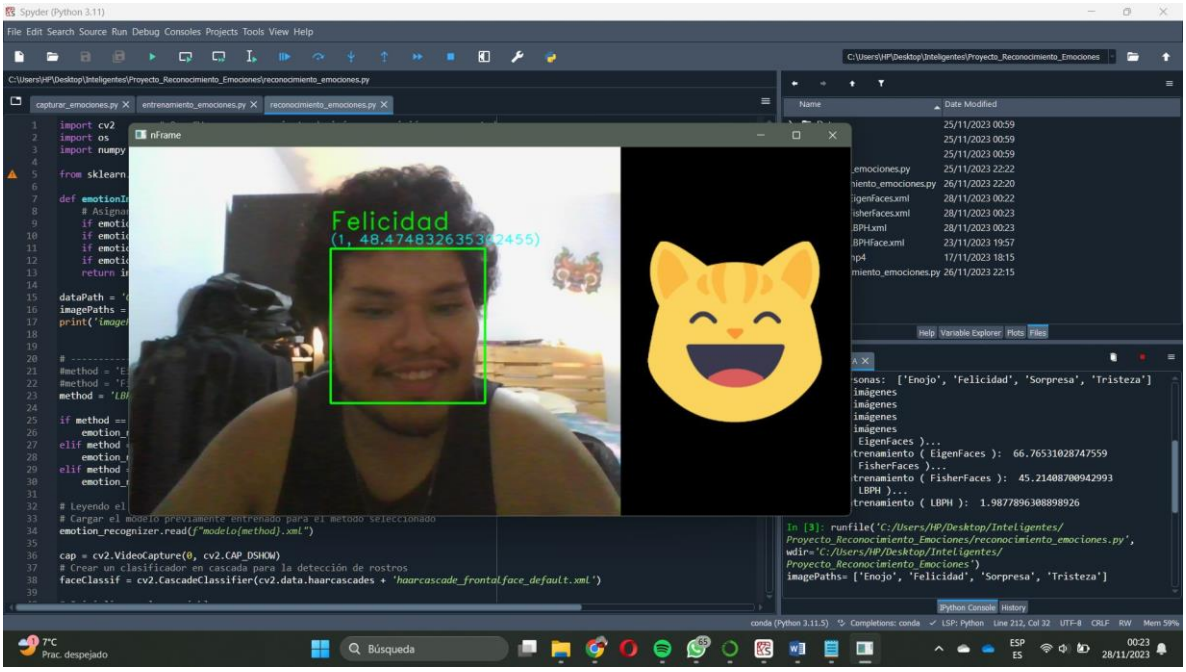
Ejemplo de Entrenamiento.

Aquí entrena las imágenes por los tres métodos.



Ejemplo de Salida.

Como salida tenemos el reconocimiento de la emoción.



Evaluación.

Métodos utilizados para evaluar el rendimiento del sistema

Para tratar de evaluar el rendimiento y efectividad del modelo se emplearon las métricas más conocidas de evaluación como la matriz de confusión para poder analizar que emoción es detectada más veces, una sensibilidad para cada emoción lo cual nos indica la proporción de casos de cada emoción que el modelo ha identificado correctamente, también una precisión para cada emoción lo cual nos indica la proporción de casos identificados, por ejemplo, casos como Felicidad que son realmente Felicidad y de igual manera se calculó el F1-score el cual combina la precisión y la sensibilidad en un solo resultado para reforzar las métricas anteriores.

El sistema no cuenta como tal con una métrica o número exacto que diga que tan efectivo es todo el modelo ya que todos los resultados dependerán del tiempo en el que se estén reconociendo las emociones y también porque no existe un conjunto de prueba donde se pueda comprar los aciertos de las predicciones con el conjunto de entrenamiento como lo hacen otros modelos ya que esa comparación es esencial para la evaluación, para tener un conjunto de pruebas tendríamos que tener dos conjuntos de datos para cada emoción haciendo que el sistema se haga un poco más complejo y

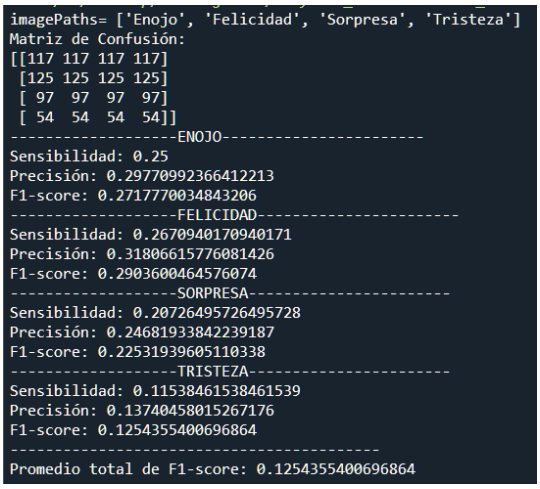
ocupe más espacio, tiempo y recursos, fue también por eso que casi todas las métricas fueron calculadas por separado para cada emoción.

Entonces para poder acercarnos un poco a lo que podría llamarse una métrica que permita conocer el grado de efectividad, logramos calcular el promedio del F1-score ya que este dato es la combinación de la sensibilidad, precisión prácticamente de todas las emociones ya que cada F1-score engloba a estos datos, donde si el F1-score está cerca de 0 quiere decir que el modelo no fue tan efectivo y si está cerca de 1 entonces si es efectivo, pero, como se dijo en un principio todo dependerá del tiempo que pase el modelo reconociendo emociones.

Debido a que el modelo trabaja con tres métodos de detección facial que son EigenFaces, FisherFaces y LBPH, se trató de ver cuál de estos es el más óptimo, pero, se llegó a la conclusión que las diferencias entre los métodos serían más cualitativas que cuantitativas, ya que lo que los diferencia es el entorno en los que se los pruebe, por ejemplo uno es más sensible a los cambios de iluminación, otro a las poses, distancias, mientras que otros dejan de lado esto y se van a los pixeles de las fotos entonces, todo depende del lugar en donde se haga la prueba el modelo, en donde sí se obtiene una diferencia significativa y podemos decir que un método es mejor que el otro es en el tiempo de entrenamiento uno tarda más que los otros.

En resumen, se calculó una matriz de confusión para saber que emoción se detectó más veces, se usó la métrica de F1-score para tratar de evaluar a todo el modelo y entre los tres métodos utilizados al momento de reconocer emociones, uno es mejor que el otro dependiendo el lugar donde se ejecute el modelo, pero, en el entrenamiento si se puede saber que método es mejor de acuerdo al tiempo que tarda en procesar todo.

Presentación y análisis de los resultados obtenidos.



Una vez concluido todo el proceso de reconocimiento, el modelo presenta las métricas mencionadas anteriormente, a continuación, se precederá a la interpretación y análisis de los resultados.

1. **Matriz de confusión.**

   La matriz de confusión de la imagen nos indica una proporción de aproximación de que emoción se registra más, se puede observar que enojo se registró en una proporción de 117, felicidad se registró 125, sorpresa 97 y Tristeza 54.

   Tambien se puede representar de una manera mas garfica y sencilla:

   ```
   [[117 117 117 117]      [[enojo    117      117      117]
    [125 125 125 125]      [ 125   felicidad   125      125]
    [97   97  97  97]      [ 97       97    sorpresa    97]
    [54   54  54  54]]     [ 54       54       54    tristeza]]
   ```

2. **Sensibilidad.**

   La sensibilidad nos indica la proporción de casos de cada emoción que el modelo ha identificado correctamente. En la imagen se registró una sensibilidad de 0.25 para enojo, 0.267 para felicidad, 0.207 para sorpresa y finalmente 0.115 para tristeza y si nos damos cuenta estos datos son proporcionales a los resultados de la matriz de confusión.

3. **Precisión.**

   Esta métrica nos indica la proporción de casos identificados por ejemplo casos como Enojo que son realmente Enojo. Los resultados nos dicen que se obtuvo una proporción de 0.297 de casos como enojo que realmente son enojo, 0.318 de felicidad que realmente es felicidad, 0.246 de sorpresa que realmente son sorpresa y 0.137 de tristeza que realmente es tristeza, como en la sensibilidad, estos resultados también son proporcionales a la matriz de confusión.

4. **F1-score.**

   Combina la precisión y la sensibilidad en una sola métrica, para poder reforzar la evaluación. Para enojo se obtuvo un score de 0.27, para felicidad 0.29, para sorpresa 0.22 y tristeza un 0.12.

   Para saber que tan tano bueno es el rendimiento mediante esta métrica se dice que si el valor es cerca de 1 indica que el modelo o mejor dicho la predicción de la emoción es óptima y si se acerca a 0 entonces no lo es, una vez aclaramos que todas estas métricas van a variar de acuerdo al tiempo en que el sistema este reconociendo rostros, por ejemplo, si se

reconoce más veces la felicidad entonces los resultados serán más óptimos para esta emoción.

5. **Promedio F1-score.**
   Debido a que cada métrica se calcula para cada emoción por separado y que el F1- score es la metica más representativa por asi decirlo del rendimiento del modelo, se decidió sacar un promedio de esta métrica para tener una idea de rendimiento en general, en el ejemplo obtuvimos un promedio de 0.125, lo que da a entender que no es óptimo, pero, hay que tener en cuenta también el tiempo, mientras más tiempo el modelo este reconociendo emociones, las métricas serán mejores.

Posibles mejoras y limitaciones identificadas.

El modelo en si está completo y bien estructurado manteniéndose en algo simple de entender y debido a esta simpleza es que se pudieron identificar algunas posibles mejoras para trabajos futuros, como, por ejemplo, en la parte del entrenamiento, la cantidad de imágenes de entrenamiento (200 por emoción) es un buen comienzo, pero la calidad y diversidad de los datos también son importantes deberíamos de tener variabilidad en términos de iluminación, fondos, expresiones faciales, etc. Dentro del entrenamiento estamos utilizando métodos clásicos como Eigenfaces, Fisherfaces y LBPH para el reconocimiento facial, estos métodos funcionar bien, pero tenemos que tener en cuenta que, para tareas más complejas y conjuntos de datos grandes, a menudo se prefieren enfoques basados en deep learning. En lo que se refiere a tercera parte donde se reconocen las emociones en el caso de Eigenfaces, Fisherfaces y LBPH, estamos utilizando umbrales fijos para decidir si una cara es reconocida o no. Se podría experimentar con diferentes umbrales para ajustar la sensibilidad y la precisión del modelo.

Conclusiones.

Hallazgos más Importantes.

Al realizar el trabajo se encontraron puntos interesantes que ayudan a los que es el reconocimiento facial y visión por computadora, algunos puntos son:

1. Deteccion y Captura de Datos: La detección de rostros se realiza utilizando el clasificador HaarCascade, y se capturan imágenes para entrenar el modelo de reconocimiento facial.
2. Entrenamiento del Modelo: Se emplean métodos clásicos de reconocimiento facial, como Eigenfaces, Fisherfaces y LBPH, para entrenar el modelo de reconocimiento de emociones. El modelo se entrena con imágenes de rostros etiquetadas con las emociones correspondientes.

3. Reconocimiento en Tiempo Real: Se implementa el reconocimiento en tiempo real de emociones utilizando el modelo entrenado y se visualizan los resultados en un fotograma de la cámara.
4. Evaluación del Rendimiento: Se evalúa el rendimiento del modelo utilizando matrices de confusión y métricas como sensibilidad, precisión y F1-score para cada emoción.

Reflexiones sobre eficacia aplicabilidad en un entorno más amplio.

El sistema proporciona una base sólida para el reconocimiento de emociones, pero hay oportunidades para mejoras y ajustes para abordar las limitaciones y hacerlo más aplicable en escenarios del mundo real.

1. Eficiencia y Precisión: La eficacia del sistema dependerá de la calidad y cantidad de datos de entrenamiento. Tenemos que asegurarnos de tener conjuntos de datos representativos y equilibrados ya que es crucial. Los métodos clásicos de reconocimiento facial son efectivos, pero su rendimiento puede variar según la complejidad del problema.
2. Limitaciones: Los métodos utilizados pueden tener limitaciones en situaciones de iluminación variable, poses extremas o variaciones en la edad y raza. El uso de umbrales fijos para la aceptación de reconocimiento puede no ser óptimo en todas las situaciones.
3. Aplicabilidad en un Entorno Más Amplio: El sistema es una implementación inicial y educativa de reconocimiento de emociones. Para entornos más amplios y desafiantes, se podría considerar el uso de enfoques basados en deep learning.

Trabajos Futuros.

Se podrían explorar mejoras adicionales, como la incorporación de modelos de deep learning, la expansión a más emociones y la optimización de la detección de rostros. La recopilación continua de datos y la actualización del modelo mejorarían la precisión con el tiempo y tal vez tratar de aplicar el modelo a empresas que requieran este servicio o algo parecido. Tal vez un Trabajo de Tesis en un futuro.

Referencias.

https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/57178/Avances%20y%20desafios%20de%20la%20inteligencia%20artificial%20-%20Serrahima%20de%20Bedoya%2C%20Alvaro.pdf?sequence=2

https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/9278/Coronel%20Caj%C3%A1n%20Erick%20Arturo.pdf?sequence=1&isAllowed=y

https://blogs.iadb.org/trabajo/es/inteligencia-artificial-que-aporta-y-que-cambia-en-el-mundo-del-trabajo/

https://arxiv.org/abs/2102.02574

https://rockcontent.com/es/blog/inteligencia-artificial-en-las-empresas/

https://biotechmagazineandnews.com/inteligencia-artificial-para-detectar-emociones/#:~:text=Tradicionalmente%2C%20la%20detecci%C3%B3n%20de%20emociones,gestos%20corporales%20o%20movimientos%20oculares.

https://culiacan.tecnm.mx/wp-content/uploads/2020/07/TesisDanielRZC.pdf