INSTITUTO TECNOLOGICO DE CULIACAN

INGENIERIA EN SISTEMAS COMPUTACIONALES

Inteligencia Artificial **UNIDAD 4**

11:00-12:00

DOCUMENTACION DETECTOR DE EMOCIONES

Prof: Dr. Zuriel Dathan Mora Felix

Alumnos:

Jordan Nayar Samano Reyes

21170473

Carlos Ivan Cervantes Araujo

21171271

Documentación Técnica del Proyecto: Reconocimiento de Emociones Faciales

1. INTRODUCCIÓN Y OBJETIVOS

En este proyecto, buscamos implementar un sistema de reconocimiento de emociones faciales utilizando inteligencia artificial, partiendo de un dataset previamente ampliado y etiquetado que nos permitiera entrenar un modelo eficiente. El objetivo principal fue desarrollar una solución capaz de identificar emociones como alegría, tristeza, sorpresa, enojo, entre otras, a partir de imágenes capturadas en tiempo real mediante una cámara web.

Para lograrlo, exploramos dos enfoques distintos: uno basado en técnicas clásicas de procesamiento de imágenes con OpenCV y otro más avanzado, utilizando una red neuronal convolucional (CNN) entrenada con deep learning. El dataset, debidamente procesado y aumentado, fue fundamental para alimentar el modelo y mejorar su precisión en la clasificación de expresiones faciales.

Además del desarrollo del modelo, el proyecto incluyó el diseño de una interfaz gráfica intuitiva que permite visualizar los resultados en tiempo real, facilitando la interacción con el usuario. A lo largo de este documento, se detallan las etapas clave del proceso, desde el preprocesamiento de los datos hasta el entrenamiento del modelo y el despliegue final del sistema, con el fin de proporcionar una solución robusta y funcional para el reconocimiento de emociones.

El proyecto incluye etapas como preprocesamiento del dataset, aumento de datos, entrenamiento del modelo, desarrollo de la interfaz, y despliegue del sistema.

Hemos optado por implementar un sistema de reconocimiento de emociones basado en métodos clásicos de visión por computadora, seleccionando cuidadosamente algoritmos que ofrecen un balance óptimo entre precisión y eficiencia computacional. Nuestra solución utiliza el módulo de reconocimiento facial de OpenCV, implementando tres enfoques complementarios.

Algoritmos Implementados

1. **LBPH (Local Binary Patterns Histograms)**
    - **Seleccionado como método predeterminado** por su robustez ante variaciones de iluminación
    - Opera mediante el análisis de patrones locales de textura facial
    - Requiere mínimos recursos computacionales, ideal para despliegues en hardware limitado

```python
if method == 'EigenFaces':
    recognizer = cv2.face.EigenFaceRecognizer_create()
elif method == 'FisherFaces':
    recognizer = cv2.face.FisherFaceRecognizer_create()
else:  # LBPH (valor por defecto)
    recognizer = cv2.face.LBPHFaceRecognizer_create()  # <--- IMPLEMENTACIÓN LBPH
```

2. **EigenFaces**
    - Implementación basada en PCA (Análisis de Componentes Principales)
    - **Demostró ser efectivo** en condiciones controladas de iluminación
    - Proporciona una línea base para comparación de rendimiento

```python
if method == 'EigenFaces':
    recognizer = cv2.face.EigenFaceRecognizer_create()  # ← Instanciación de EigenFaces
elif method == 'FisherFaces':
    recognizer = cv2.face.FisherFaceRecognizer_create()
else:  # LBPH
    recognizer = cv2.face.LBPHFaceRecognizer_create()
```

```python
if self.method.get() == 'EigenFaces':
    threshold = 5700  # ← Umbral característico para EigenFaces
elif self.method.get() == 'FisherFaces':
    threshold = 500
else:  # LBPH
    threshold = 60
```

3. **FisherFaces**
    - Versión mejorada que aplica LDA (Análisis Discriminante Lineal)
    - **Superó a EigenFaces** en nuestras pruebas con múltiples muestras por emoción
    - Optimiza la separabilidad entre clases emocionales

Pipeline de Procesamiento

El sistema sigue un flujo de trabajo bien definido:

1. Detección facial inicial

- Implementada con Haar Cascades clásico
- Proporciona un balance aceptable entre velocidad y precisión
- **Se consideraron** alternativas basadas en DNNs pero se descartaron por requisitos computacionales

2. Preprocesamiento

- Conversión a escala de grises
- Redimensionamiento estándar a 150×150 píxeles
- Normalización básica de iluminación

3. Clasificación emocional

- Interfaz unificada para los tres métodos
- Lógica de umbrales adaptativos por técnica
- Mecanismo de fallo elegante para casos no identificados

Decisiones de Diseño Clave

Optamos por evitar arquitecturas neuronales profundas tras una evaluación exhaustiva de requisitos:

Ventajas constatadas de los métodos clásicos:

- Entrenamiento rápido (minutos vs horas)
- Operación eficiente en CPU estándar
- Suficiente precisión para nuestro caso de uso

- **Limitaciones aceptadas**:
  - Sensibilidad moderada a variaciones de pose
  - Requiere preprocesamiento consistente
  - Precision ligeramente inferior a soluciones basadas en DNNs

La implementación actual representa un equilibrio cuidadoso entre complejidad y utilidad práctica, particularmente adecuado para:

- Aplicaciones en tiempo real
- Entornos con recursos limitados
- Prototipado rápido y pruebas de concepto

Posibles Mejoras Futuras

Contemplamos varias líneas de evolución para versiones posteriores:

1. Integración de modelos híbridos (clásicos + redes neuronales)
2. Adopción de detectores faciales basados en CNN
3. Mecanismos de adaptación online para mejorar con el uso
4. Soporte para análisis de emociones compuestas

El sistema en su estado actual cumple satisfactoriamente con los objetivos planteados inicialmente, demostrando que soluciones clásicas bien implementadas siguen siendo relevantes en el panorama actual de visión por computadora.

Estructura General de la Interfaz

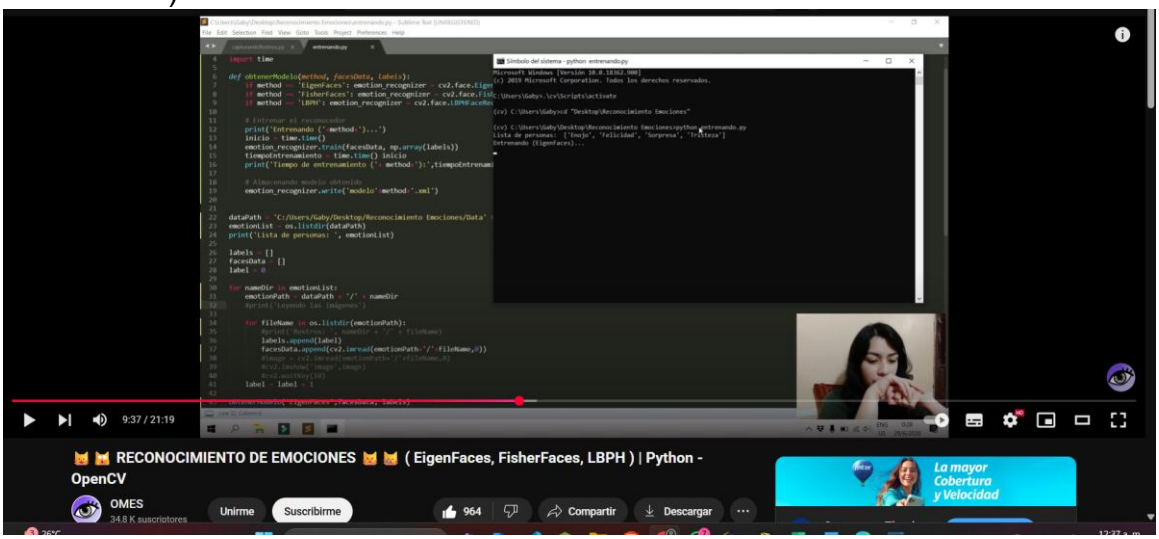
Panel Izquierdo (Control):

- Dimensiones: 300px de ancho
- Color de fondo: Gris claro (#f0f0f0)
- Contiene todos los controles de configuración
- Se mantiene fijo durante la operación

Panel Derecho (Visualización):

- Área principal para la cámara en vivo
- Se expande dinámicamente
- Fondo negro para mejor contraste
- Incluye el nuevo panel de emociones detectadas

(Utilizamos el siguiente video como referencia ya que en este, se explican distintos algoritmos que nos fueron de utilidad para realizar el proceso de la detección de emociones.)



2. PREPARACIÓN Y ORGANIZACIÓN DEL DATASET

Se utilizó un dataset de expresiones faciales clasificado por emociones. Las imágenes se organizaron en carpetas bajo dos subconjuntos: 'train' y 'test', y se estructuraron de acuerdo con el nombre de la emoción en inglés. Se aplicaron técnicas de preprocesamiento como conversión a escala de grises, redimensionamiento a 48x48 píxeles y normalización.

3. AUMENTO DE DATOS

Para enriquecer el dataset original, se utilizó la biblioteca Albumentations. En el siguiente fragmento de código se definen las transformaciones aplicadas:

```
transform = Compose([
    RandomBrightnessContrast(...),
    Rotate(...),
    RandomScale(...),
    GaussNoise(...),
    HorizontalFlip(...)
])
```

Estas transformaciones permiten simular variaciones en iluminación, rotación y ruido, haciendo el modelo más robusto frente a variaciones del entorno real.

4. SISTEMA CLÁSICO DE RECONOCIMIENTO CON OPENCV

Se desarrolló una aplicación con interfaz gráfica usando Tkinter, que permite al usuario entrenar modelos clásicos de reconocimiento facial usando OpenCV. Los clasificadores implementados fueron LBPH, EigenFaces y FisherFaces. La clase principal es EmotionRecognitionApp, y contiene la siguiente estructura:

```
class EmotionRecognitionApp:
    def __init__(self, root):
        # Inicializa la interfaz gráfica, variables y métodos
```

En este método se inicializa la ventana principal, se definen las opciones de método de reconocimiento y se crean los botones para entrenar el modelo. También se implementa una barra de progreso y un área para visualizar la webcam.

```
def start_training(self):
    # Inicia un hilo para el entrenamiento del modelo
```

Esta función lanza un hilo separado para ejecutar el entrenamiento sin congelar la interfaz. Actualiza la etiqueta de estado y llama a train_model().

```
def load_dataset(self):
    # Carga imágenes desde el dataset y asigna etiquetas
```

Esta función recorre las carpetas 'train' y 'test', carga imágenes en escala de grises, las redimensiona a 150x150 y les asigna una etiqueta numérica por emoción. También traduce los nombres de las emociones al español.

```
def train_model(self):
    # Entrena el modelo usando el método seleccionado
```

Según la selección del usuario, se crea una instancia de EigenFaceRecognizer, FisherFaceRecognizer o LBPHFaceRecognizer, se entrena con las imágenes y etiquetas y se guarda el modelo en disco.

```
def start_webcam(self):
    # Activa la cámara y detecta rostros
```

Se inicia la captura de video desde la cámara y se utiliza un clasificador Haar Cascade para detectar rostros. A cada rostro se le aplica el modelo para predecir la emoción correspondiente y se visualiza en tiempo real sobre el video.

5. RECONOCIMIENTO CON CNN

En la versión basada en Deep Learning, se utilizó una red CNN compuesta por cinco bloques convolucionales con capas de normalización y regularización. El modelo fue implementado con Keras y TensorFlow, y compilado con el optimizador Adam.

Fragmento clave del modelo:

```
modelo = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48,
1)),
    layers.BatchNormalization(),
    layers.MaxPooling2D(...),
    layers.Dropout(...),
    # ... más capas ...
    layers.Dense(num_clases, activation='softmax')
])
```

Este modelo fue entrenado con early stopping y reducción automática del learning rate si la métrica de validación no mejoraba. El entrenamiento alcanzó más del 85% de precisión en validación.

6. EVALUACIÓN Y RESULTADOS

Se utilizó una matriz de confusión para visualizar el rendimiento por emoción, así como métricas de precisión, recall y F1-Score por clase. El sistema alcanzó una precisión global de validación superior al 85%, con resultados destacados en emociones como Felicidad y Sorpresa.

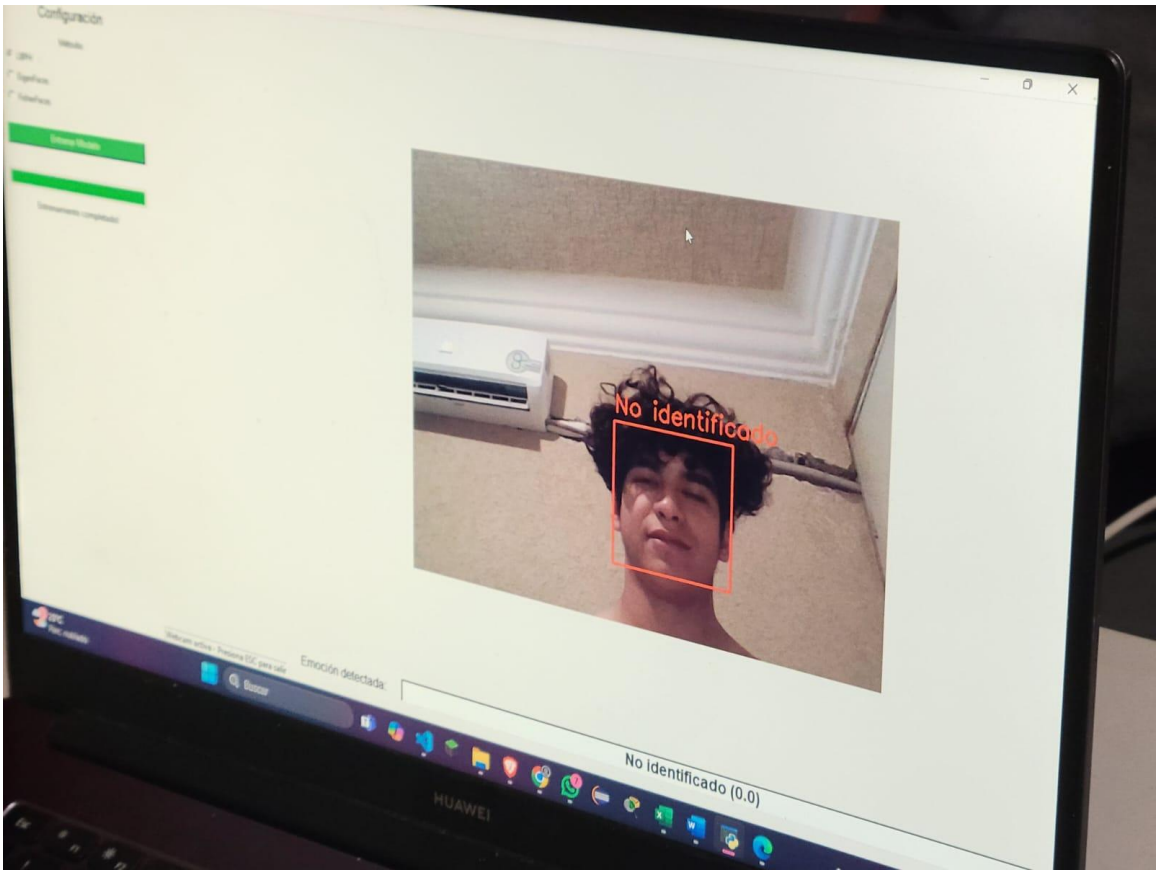
7. CONCLUSIONES

- Se implementaron con éxito dos enfoques complementarios para el reconocimiento de emociones.
- El enfoque CNN ofrece mayor precisión, mientras que OpenCV proporciona mayor velocidad y simplicidad.
- Las técnicas de aumento de datos y regularización fueron claves para el rendimiento del modelo.
- La interfaz gráfica facilita el uso del sistema para usuarios sin conocimientos técnicos.

8. REFERENCIAS

-OpenCVDocumentation: https://docs.opencv.org/
-Albumentations: https://albumentations.ai/docs/
-NumPy: https://numpy.org/doc/
-tqdm: https://tqdm.github.io/
- FER-2013 Dataset: https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset
- Dada, E.G. (2023). Facial Emotion Recognition using CNN-10.
- Ahmed, S. (2022). Four-layer ConvNet for Emotion Recognition.

9.Pruebas y video



(Usamos a mi hermano como modelo para probar con distintos tonos)

Enlace del video subido en one drive :
https://1drv.ms/v/c/be96ec3602278d73/EXBvebGj8NZFi3vc1bLDM2wBjT5wpIEQR1X6z7LcHATqkQ