



TECNOLÓGICO
NACIONAL DE MÉXICO



INSTITUTO TECNOLÓGICO DE CULIACÁN

CARRERA

ING. EN SISTEMAS COMPUTACIONALES

MATERIA

INTELIGENCIA ARTIFICIAL

UNIDAD 4

TAREA 2

ALUMNOS

TRUJILLO ACOSTA BRYANT

BRAYAN MENDOZA GARCIA

PROFESOR

MORA FÉLIX ZURIEL DATHAN

Primeramente para este proyecto de reconocimiento de emociones mediante la camara web, nos dimos la tarea de investigar la arquitectura neuronal recomendable por la comunidad científica, la que nosotros decidimos elegir y que era mas recomendable fue red neuronal convolucional con Keras.

Las redes neuronales convolucionales consiste en aplicar redes neuronales sobre imágenes. Clasificar imágenes, detectar qué contienen, generar nuevas imágenes... todo esto es posible mediante redes neuronales convolucionales.

Las redes neuronales convolucionales han sido una de las innovaciones más influyentes en el campo de la visión por ordenador. Han funcionado mucho mejor que la visión por ordenador tradicional y han producido resultados de vanguardia. Estas redes neuronales han demostrado su eficacia en muchos estudios de casos y aplicaciones de la vida real, como:

- Clasificación de imágenes, detección de objetos, segmentación, reconocimiento facial.
- Coches autoconducidos que aprovechan los sistemas de visión basados en CNN.
- Clasificación de la estructura cristalina mediante una red neuronal convolucional.

Keras viene con una biblioteca llamada datasets, que puedes utilizar para cargar conjuntos de datos nada más sacarlos de la caja: descargas los datos del servidor y aceleras el proceso, puesto que ya no tienes que descargar los datos a tu ordenador. Las imágenes de entrenamiento y de prueba, junto con las etiquetas, se cargan y almacenan en las variables.

El reconocimiento de emociones faciales es un área dentro de la visión por computadora donde se usan redes neuronales convolucionales (CNN) debido a su eficacia para aprender características espaciales de las imágenes faciales.

PARAMETROS DE ENTRENAMIENTO

Parámetro	Recomendación general	Explicación
Epochs	15–50	Más si tienes datos balanceados. Usa early stopping.
Batch size	32–64	Valores mayores consumen más RAM pero mejoran estabilidad del gradiente.
Learning rate	0.001 (default Adam)	Usa ReduceLROnPlateau si la val_loss se estanca.
Optimizador	Adam (recomendado)	Rápido, estable y muy usado en reconocimiento facial.
Loss function	categorical_crossentropy	Para clasificación multiclas.
Regularización	Dropout (0.3–0.5) o L2 regularization	Evita sobreajuste.
Augmentación	Sí	Reflejos horizontales, rotación, zoom. Mejora la generalización.
Validation split	0.1–0.2	Usa parte del conjunto de entrenamiento para validación.

Explicacion del programa

primeramente la explicacion del programa para hacer el entrenamiento se usaron las imágenes ya preprocesadas que se hicieron en la tarea anterior.

Importación de librerías

```

1 import cv2
2 import os
3 import time
4 import numpy as np
5 import tensorflow as tf
6 from tensorflow import keras
7 from tensorflow.keras.models import Sequential
8 from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

```

Usamos tensorflow y como ya habiamos investigado usamos el modelo keras que es el mas recomendado para el modelo que queremos hacer para detectar emociones a traves de entrenar nuestro sistema con imágenes.

Definición de la clase EmotionRecognizer

```
class EmotionRecognizer:  
    def __init__(self):  
        self.emotions = ['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']  
        self.IMG_SIZE = 48  
  
        self.model = self.build_model()
```

- Define la lista de emociones .
- IMG_SIZE = 48 es el tamaño estándar de las imágenes.
- Llama a self.build_model() para construir la red neuronal.

build_model: Construye la red neuronal convolucional

```
def build_model(self):  
    """Construye el modelo CNN"""  
    model = Sequential()  
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(self.IMG_SIZE, self.IMG_SIZE, 1)))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
  
    model.add(Conv2D(64, (3, 3), activation='relu'))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
  
    model.add(Flatten())  
    model.add(Dense(128, activation='relu'))  
    model.add(Dropout(0.5))  
    model.add(Dense(len(self.emotions), activation='softmax'))  
  
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])  
    return model
```

- Dos capas convolucionales (Conv2D) con activación ReLU y max pooling para reducción espacial.
- Flatten() convierte los mapas de características en un vector.
- Dense(128) es una capa totalmente conectada.
- Dropout(0.5) reduce el sobreajuste.
- Dense(len(self.emotions), activation='softmax') da la predicción final con 7 salidas (una por emoción).
- Se compila el modelo usando Adam y categorical_crossentropy, ya que es una clasificación multiclas.

train_model: Carga las imágenes y entrena el modelo

```
def train_model(self, augmented, epochs=15, batch_size=32):
```

- augmented: carpeta raíz donde están las subcarpetas de emociones con las imágenes aumentadas.
- epochs y batch_size definen el entrenamiento.

Leer las imágenes

```
for emotion_idx, emotion in enumerate(self.emotions):  
    emotion_path = os.path.join(augmented, emotion)  
    if not os.path.exists(emotion_path):  
        print(f"Carpetas no encontradas: {emotion_path}")  
        continue
```

Recorre cada carpeta por emoción. emotion_idx se convierte en la etiqueta numérica.

Lee, redimensiona y normaliza las imágenes (a valores entre 0 y 1).

```
    img_resized = cv2.resize(img, (self.IMG_SIZE, self.IMG_SIZE))  
    img_normalized = img_resized.astype('float32') / 255.0  
    X_train.append(img_normalized)  
    y_train.append(emotion_idx)  
except Exception as e:  
    print(f"Error procesando {file}: {e}")
```

Guarda la imagen en X_train y su clase en y_train.

Preparar los datos para el modelo

```
X_train = np.array(X_train)  
y_train = keras.utils.to_categorical(y_train, num_classes=len(self.emotions))  
X_train = np.expand_dims(X_train, -1)
```

- Convierte X_train a un numpy array.
- y_train se convierte a one-hot encoding.
- expand_dims agrega la dimensión de canales (porque son imágenes en escala de grises → 1 canal).

Entrenar y guardar modelo

```
print("Entrenando modelo ...")
self.model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_split=0.1)
```

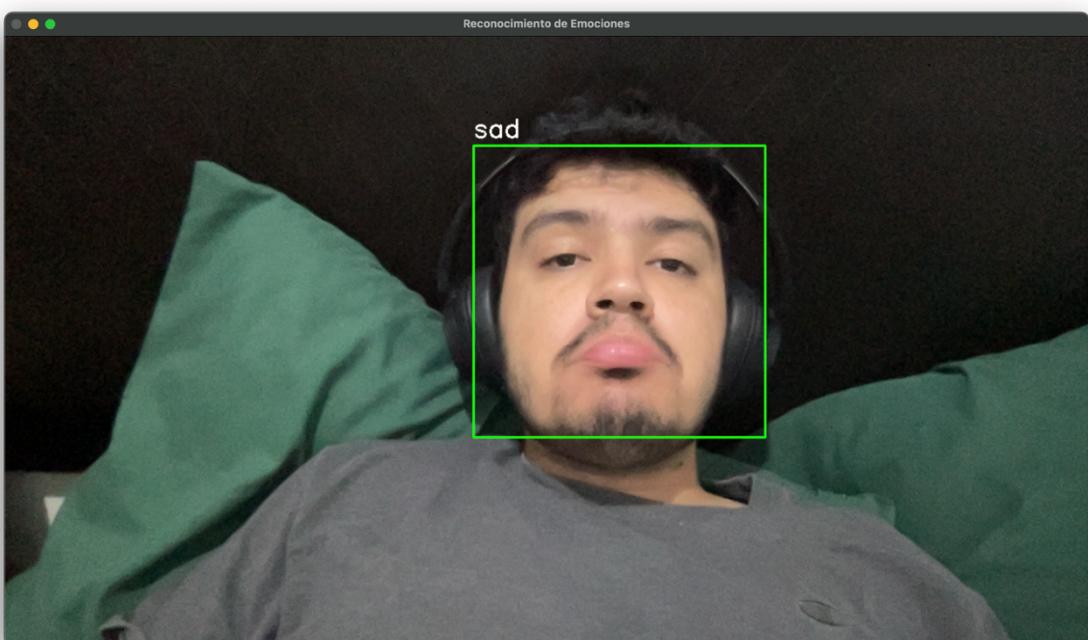
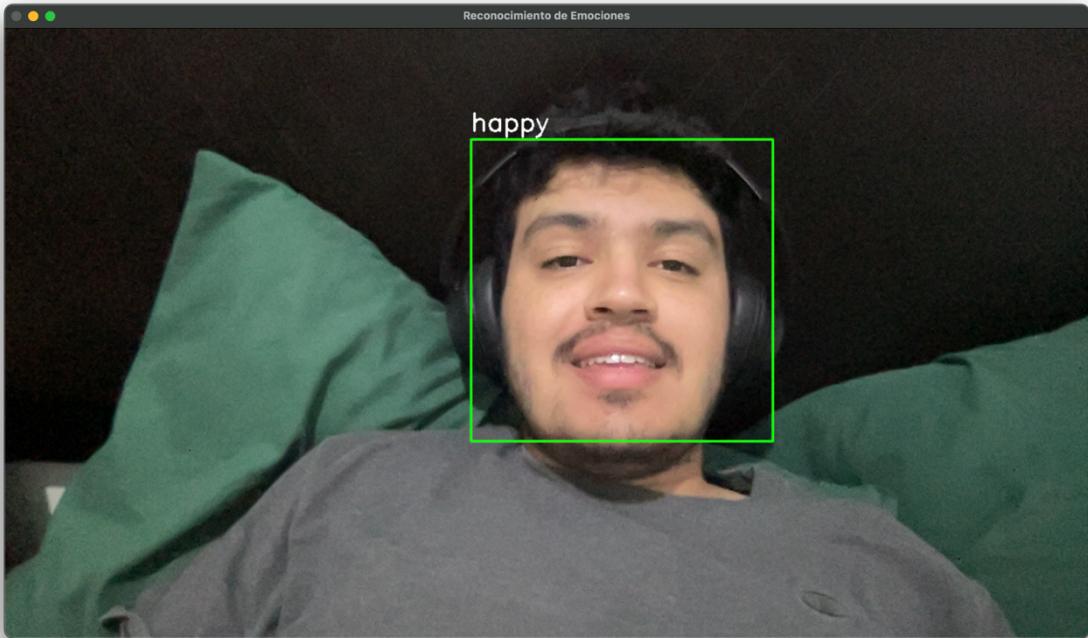
Entrena el modelo con 10% de los datos reservados para validación.

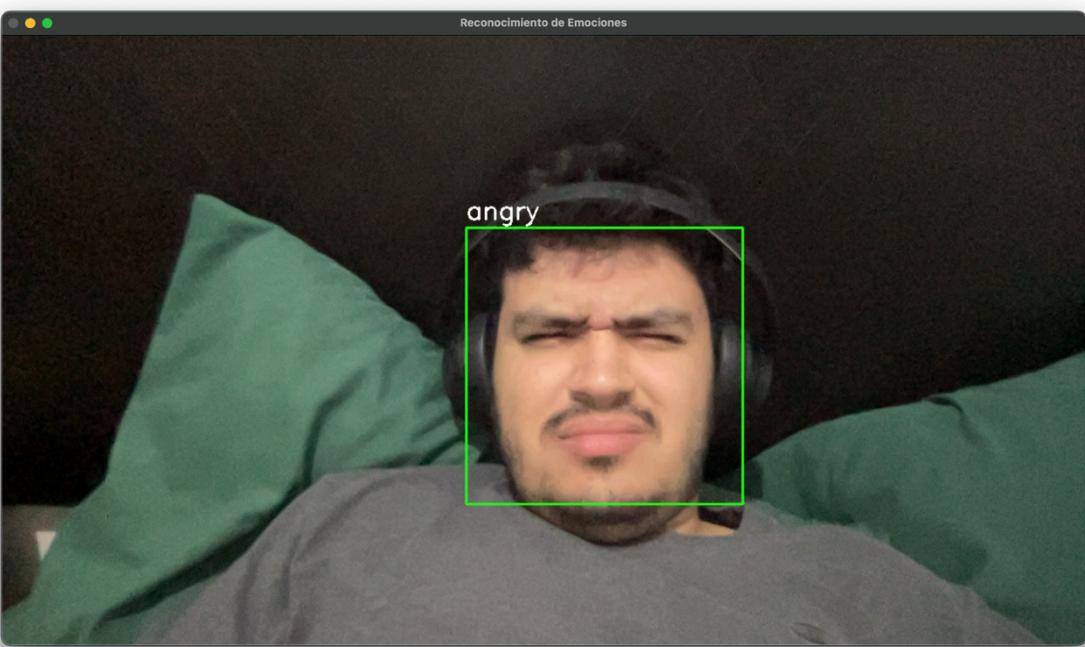
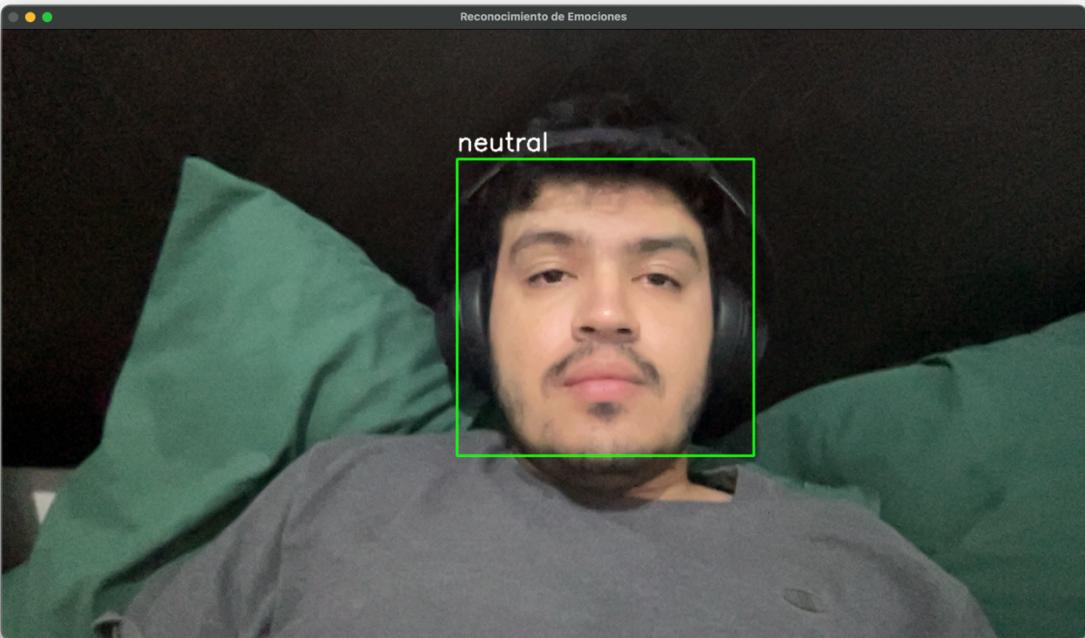
Bloque principal: ejecuta el entrenamiento

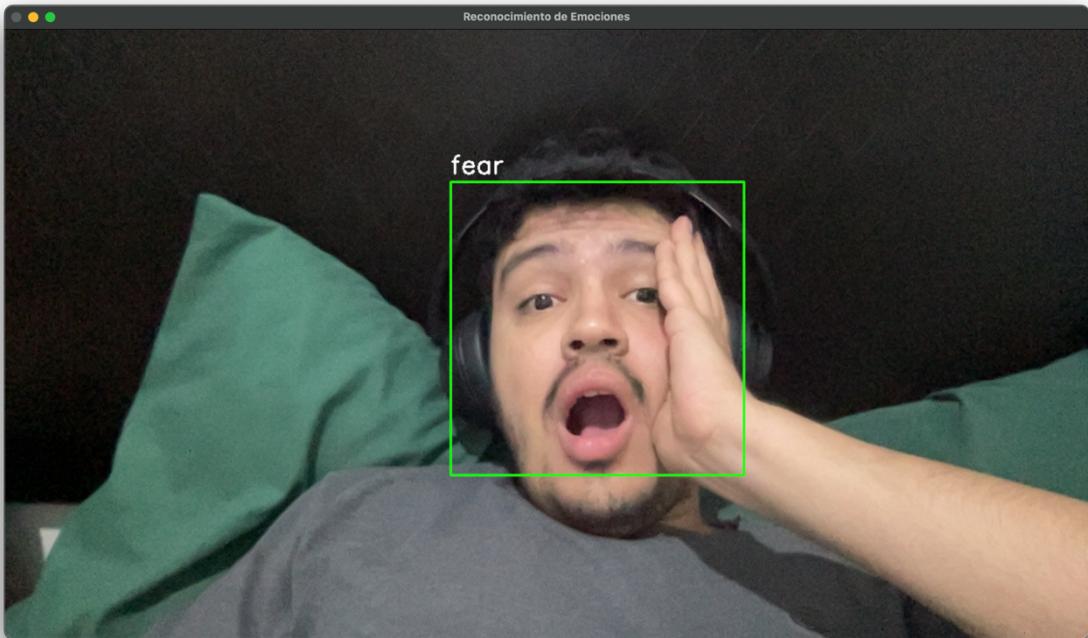
```
if __name__ == "__main__":
    recognizer = EmotionRecognizer()
    recognizer.train_model("augmented")
```

- Crea una instancia de la clase EmotionRecognizer.
- Llama a train_model() usando como carpeta "augmented" (debe tener subcarpetas como augmented/angry, augmented/happy, etc. con imágenes dentro).esto se hizo asi porque nuestro dataset viene dividido por carpetas dependiendo la emocion

CAPTURAS DE PANTALLA DEL PROGRAMA REAL







LINK DEL VIDEO DE YOUTUBE USANDO LA APLICACIÓN

<https://youtu.be/xeMKmVmj-7g>

Bibliografía

<https://www.datacamp.com/es/tutorial/convolutional-neural-networks-python>
<https://anderfernandez.com/blog/que-es-una-red-neuronal-convolucional-y-como-crearlaen-keras/>
https://www.kaggle.com/datasets/msambare/fer2013/data