Tecnológico Nacional de México Campus Culiacán

TECNOLÓGICO NACIONAL DE MÉXICO

"Documentación-Detector de emociones"

Nombre de los alumnos:

Rios Sauceda Jose Lorenzo.

Cazarez Ibarra Francisco Javier

Docente: *Zuriel Dathan Mora Félix.*

Materia: *Inteligencia Artificial.*

Carrera: *Ing. En sistemas computacionales.*

Semestre: *8*

Horario: *9-10AM*

Fecha: *29 de mayo del 2025.*

¿Qué tipo de red neuronal se utilizó?

Se utilizó una Red Neuronal Convolucional (CNN) en este trabajo porque es la arquitectura más adecuada para el procesamiento de imágenes, como las capturas faciales necesarias para detectar emociones que fueron requeridas para este trabajo. Son robustas ante variaciones en la posición, escala o expresión del rostro, lo que mejora la precisión del sistema. Por su eficiencia y efectividad comprobada en tareas de visión por computadora, creemos que fue ideal trabajar con ella en este proyecto.

Parámetros de entrenamiento

Épocas (epochs): 20 Esto representa y define el numero de veces que el modelo que definimos tiene que recorrer el dataset.

Tamaño del lote (batch_size): 32 Define el número de imágenes que serán procesadas antes de actualizar los pesos que maneja el modelo.

Optimizador: Adam Este algoritmo es de optimización que se adapta de manera amplia, es muy utilizado en CNN.

Función de pérdida: categorical_crossentropy Se utilizo para la clasificación multiclase de emociones.

Métrica: accuracy Esto nos ayuda a medir que tan frecuentemente se presentan predicciones que son correctas.

Enlace del video del funcionamiento del código: https://youtu.be/qsRIshncFkw

Importante:

Decidimos cambiar a MobileNetV2 porque es un modelo más ligero y eficiente en cuanto a recursos, lo que permite un entrenamiento y una inferencia más rápidos. A diferencia de ResNet50, MobileNetV2 está optimizado para dispositivos con menos capacidad de cómputo, ideal para aplicaciones en tiempo real como lo es esta tarea de detección de emociones por webcam. Además, al usar transferencia de aprendizaje con pesos preentrenados y técnicas como global average pooling, regularización y dropout, buscamos mejorar la generalización y evitar el sobreajuste, adaptando el modelo a nuestro conjunto de datos específico de manera más eficiente.

```
#Modelo con MobileNetV2
base_model = MobileNetV2(input_shape=input_shape, include_top=False, weights='imagenet')
base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.001))(x)
x = Dropout(0.5)(x)
predictions = Dense(num_clases, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Callbacks para mejorar el entrenamiento
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6, verbose=1)
early_stop = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True)
```

Código para la detección por webcam

Se realizan las importaciones necesarias, cv2 para procesamiento de imágenes y video, numpy para matrices y load_model de keras para cargar el modelo que ya fue entrenado. Además de que se carga el modelo preentrenado que estaremos usando para predecir emociones. También se definen las etiquetas de las salidas que tiene

el modelo creado y por ultimo se utiliza el clasificador haar de forma auxiliar para detectar rostros cada ciertos frames.

```python
import cv2
import numpy as np
from tensorflow.keras.models import load_model    Import "tensorflow.keras.models" could not be resolve

# Cargar modelo entrenado
model = load_model('modelo_emociones_transferlearning.h5')

# Clases del dataset
clases = ['Angustiada', 'Enojo', 'Feliz', 'Sorprendida', 'Triste']

# Auxiliar en el clasificador de rostros
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
```

Aquí están las variables para la frecuencia de la predicción, el contador de frames, el intervalo que indica que la predicción se hará cada 20 frames para reducir la frecuencia y mejorar la estabilidad y las ultimas dos variables que guardan el resultado de la última predicción para mostrarla si no se actualiza.

```python
# Variables para controlar frecuencia de predicción
contador_frames = 0
intervalo_prediccion = 20  # Aumentar valor = predicciones más espaciadas
ultima_etiqueta = ""
ultima_probabilidad = 0.0
```

Ahora entramos en este bucle que lee frame a frame el video de la cámara, en caso de que la cámara no se pueda visualizar se sale del bucle inmediatamente. Se detectan rostros a escala de grises porque asi lo requiere haar y para cada rostro detectado por la cámara se define un margen para capturar mejorar la expresión de la persona, luego se realiza la predicción con el modelo, se obtiene la clase con mayor probabilidad y se almacena con su respectiva etiqueta y probabilidad de este modo se dibuja un rectángulo verde alrededor del rostro y se muestra un texto

con la ultima etiqueta (emoción detectada en el rostro) y la probabilidad hasta que cambie a otra predicción, finalmente se incrementa el contador de frames por cada iteración. Para terminar la visualización en la cámara simplemente se tiene que presionar Q

```python
while True:
    ret, frame = cap.read()
    if not ret:
        break

    gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rostros = face_cascade.detectMultiScale(gris, 1.1, 4)

    for (x, y, w, h) in rostros:
        margen = int(0.1 * w)
        x1 = max(0, x - margen)
        y1 = max(0, y - margen)
        x2 = min(frame.shape[1], x + w + margen)
        y2 = min(frame.shape[0], y + h + margen)

        if contador_frames % intervalo_prediccion == 0:
            rostro = frame[y1:y2, x1:x2]
            rostro = cv2.cvtColor(rostro, cv2.COLOR_BGR2RGB)
            rostro = cv2.resize(rostro, (224, 224))
            rostro = rostro.astype('float32') / 255.0
            rostro = np.expand_dims(rostro, axis=0)

            preds = model.predict(rostro, verbose=0)
            clase_idx = np.argmax(preds)
            ultima_etiqueta = clases[clase_idx]
            ultima_probabilidad = preds[0][clase_idx]

        contador_frames += 1
```