TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CULIACÁN

ING. EN SISTEMAS COMPUTACIONALES

MODULO 4

TAREA 2

ALUMNO:

CHRISTIAN NATANAEL AYON ZAZUETA

MATERIA:

INTELIGENCIA ARTIFICIAL

MAESTRO:

ZURIEL DATHAN MORA FELIX

HORARIO

9:00-10:00

1. Introducción

Esta tarea consiste en el desarrollo de un sistema capaz de detectar emociones faciales en tiempo real a partir de imágenes capturadas con una webcam. El objetivo principal es reconocer expresiones faciales humanas y clasificarlas en siete emociones básicas: enojo, asco, miedo, felicidad, neutralidad, tristeza y sorpresa.

El sistema se basa en una red neuronal convolucional entrenada con el conjunto de datos FER2013, e implementado utilizando Python, TensorFlow y OpenCV.

2. Herramientas Utilizadas

- **Lenguaje:** Python
- **Bibliotecas:**
    - TensorFlow / Keras
    - OpenCV
    - NumPy
- **Entorno de desarrollo:** Kaggle Notebooks, Visual Studio Code

3. Arquitectura del Modelo

El modelo fue construido con Sequential de Keras. Incluye:

- Tres bloques de capas Conv2D, BatchNormalization, MaxPooling2D y Dropout
- Capas dense finales con activaciones relu y softmax
- Regularización L2 y Dropout para evitar overfitting

La función de pérdida utilizada fue categorical crossentropy, con el optimizador Adam.

```
model = Sequential([
    Input(shape=(128, 128, 1)),
    Conv2D(32, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(NUM_CLASSES, activation='softmax')
])
```

```
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

4. Entrenamiento

Se entrenó el modelo con 20 épocas, utilizando validación con el conjunto /test. Se aplicó la técnica de EarlyStopping para evitar un sobre entrenamiento innecesario.

```
early_stop = EarlyStopping(monitor='accuracy', patience=5, restore_best_weights=True)

model.fit(
    train_generator,
    epochs=20,
    callbacks=[early_stop]
)

os.makedirs('model', exist_ok=True)
model.save('/kaggle/working/modelo_emociones.h5')
```

5. Implementación

Para la detección facial, se utilizaron los clasificadores Cascade incluidos en OpenCV, que permiten localizar rostros dentro de cada fotograma capturado por la webcam.
Cada región detectada se redimensiona, normaliza y se pasa al modelo entrenado para predecir la emoción correspondiente.

```
detector_rostros = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

cam = cv2.VideoCapture(0)

while True:
    ret, frame = cam.read()
    if not ret:
        break

    gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rostros = detector_rostros.detectMultiScale(gris, scaleFactor=1.3, minNeighbors=5)

    for (x, y, w, h) in rostros:
        rostro = gris[y:y+h, x:x+w]
        rostro_redim = cv2.resize(rostro, IMG_SIZE)
        rostro_redim = rostro_redim.astype("float32") / 255.0
        rostro_redim = np.expand_dims(rostro_redim, axis=-1)  # (48, 48, 1)
        rostro_redim = np.expand_dims(rostro_redim, axis=0)   # (1, 48, 48, 1)

        pred = modelo.predict(rostro_redim, verbose=0)
        emocion = EMOCIONES[np.argmax(pred)]

        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame, emocion, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,
                    0.9, (255, 255, 255), 2)

    cv2.imshow("Detector de Emociones", frame)
```

6.- Resultados

https://youtu.be/cClf9dX0fdA

7.-Conclusion

Si bien el sistema es capaz de cumplir de manera aceptable con la detección de emociones básicas al momento de realizar las pruebas se tiene la dificultad de detectar el miedo y el asco, creo que todavía se puede mejorar aun mas si se entrena con mas datos claro sin llegar al sobre entrenamiento para esto habría que añadir un generador de validación para poder comprobar que el modelo si aprende. Dicho esto, creo que aun así es un trabajo decente.