TENSORFLOW

1. **¿Qué es TensorFlow?**

   Es una biblioteca de Machine Learning desarrollada por Google que permite construir y entrenar redes neuronales.

2. **¿Qué es Keras?**

   Es una API de alto nivel que se ejecuta sobre TensorFlow y facilita la creación de modelos de IA de forma sencilla. TensorFlow es el motor, y Keras es el "volante" que hace fácil conducirlo.

3. **Flujo de trabajo en IA**
   - Preparar datos.
   - Definir el modelo (red neuronal).
   - Entrenar el modelo.
   - Evaluar resultados.

EJERCICIOS

Ejercicio 1

Implementa el siguiente algoritmo.

4_TensorFlow_Keras > T001_prueba.py > ...

```python
# pip install tensorflow
import tensorflow as tf
import numpy as np

# ========= 1) Crear datos =========
# x de 0 a 10
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], dtype=float)
# y = 2x - 1
y = 2 * x - 1

# ========= 2) Definir el modelo =========
# Una red neuronal con 1 neurona y 1 entrada
modelo = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])
])

# ========= 3) Compilar el modelo =========
# sgd = descenso del gradiente, mse = error cuadrático medio
modelo.compile(optimizer='sgd', loss='mean_squared_error')

# ========= 4) Entrenar =========
print("Entrenando el modelo...")
historial = modelo.fit(x, y, epochs=200, verbose=0)

# ========= 5) Predicción =========
nueva_x = np.array([[10.0]])   # debe ser array 2D
prediccion = modelo.predict(nueva_x)
print("Predicción para x=10:", prediccion)
```

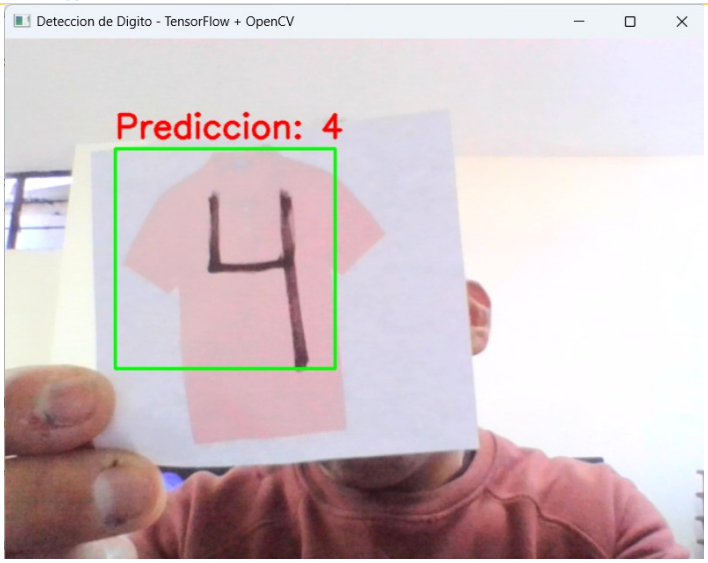
Ejercicio 2

Implementa el siguiente algoritmo.

4_TensorFlow_Keras > T002_detector_numero.py > ...

```python
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

# ====== 1) Entrenar un modelo simple con MNIST ======
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalizar y adaptar formato
x_train = x_train.reshape(-1, 28, 28, 1).astype("float32") / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype("float32") / 255.0

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D((2,2)),
    Flatten(),
    Dense(100, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, validation_data=(x_test, y_test))

# ====== 2) Capturar imagen con OpenCV ======
cap = cv2.VideoCapture(0)
```

```python

print("Presiona 'q' para salir...")

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

    # Convertir a escala de grises
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Dibujar un rectángulo donde escribir el número
    cv2.rectangle(frame, (100,100), (300,300), (0,255,0), 2)
    roi = gray[100:300, 100:300]

    # Preprocesar ROI para el modelo
    img_resized = cv2.resize(roi, (28,28))
    img_resized = cv2.bitwise_not(img_resized)  # invertir: fondo negro, número blanco
    img_resized = img_resized.astype("float32") / 255.0
    img_resized = img_resized.reshape(1,28,28,1)

    # Predicción con TensorFlow
    prediction = model.predict(img_resized, verbose=0)
    digit = np.argmax(prediction)

    # Mostrar resultado en pantalla
    cv2.putText(frame, f"Prediccion: {digit}", (100,90), cv2.FONT_HERSHEY_SIMPLEX,
                1, (0,0,255), 2, cv2.LINE_AA)

    cv2.imshow("Deteccion de Digito - TensorFlow + OpenCV", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```



Ejercicio 3

Implementa el siguiente algoritmo.

4_TensorFlow_Keras > T003_asistencia_face.py > ...

```python
import cv2
import numpy as np
import face_recognition
import tensorflow as tf
import os
import csv
from datetime import datetime

# ====== 1) Cargar imágenes de los estudiantes ======

path = r"C:\Users\DELL\Desktop\python\senati\4_TensorFlow_Keras\estudiantes"
#path = "estudiantes"  # Carpeta con fotos de estudiantes
images = []
nombres = []

for archivo in os.listdir(path):
    img = cv2.imread(f"{path}/{archivo}")
    images.append(img)
    nombres.append(os.path.splitext(archivo)[0])  # nombre = archivo sin extensión

# Convertir a codificaciones faciales
def codificar_imagenes(imgs):
    lista_codigos = []
    for img in imgs:
        rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        codigos = face_recognition.face_encodings(rgb)[0]
        lista_codigos.append(codigos)
    return lista_codigos

print("Codificando rostros de estudiantes...")
codigos_conocidos = codificar_imagenes(images)
print("¡Rostros codificados!")

# ====== 2) Crear archivo de asistencia ======
archivo_asistencia = r"C:\Users\DELL\Desktop\python\senati\4_TensorFlow_Keras\asistencia.csv"
#archivo_asistencia = "asistencia.csv"
with open(archivo_asistencia, "w", newline='') as f:
    escritor = csv.writer(f)
    escritor.writerow(["Nombre", "Fecha", "Hora"])

# ====== 3) Iniciar cámara ======
cap = cv2.VideoCapture(0)

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

```

```python
    # Redimensionar para acelerar
    img_peq = cv2.resize(frame, (0,0), None, 0.25, 0.25)
    img_rgb = cv2.cvtColor(img_peq, cv2.COLOR_BGR2RGB)

    # Detectar rostros
    caras = face_recognition.face_locations(img_rgb)
    codigos = face_recognition.face_encodings(img_rgb, caras)

    for codigo, cara in zip(codigos, caras):
        # Comparar con rostros conocidos
        matches = face_recognition.compare_faces(codigos_conocidos, codigo)
        distancias = face_recognition.face_distance(codigos_conocidos, codigo)
        idx = np.argmin(distancias)

        if matches[idx]:
            nombre = nombres[idx].upper()

            # Escalar coordenadas de cara detectada
            y1, x2, y2, x1 = cara
            y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
            cv2.rectangle(frame, (x1,y1), (x2,y2), (0,255,0), 2)
            cv2.putText(frame, nombre, (x1,y1-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)

            # Registrar asistencia
            now = datetime.now()
            fecha = now.strftime("%Y-%m-%d")
            hora = now.strftime("%H:%M:%S")

            with open(archivo_asistencia, "a", newline='') as f:
                escritor = csv.writer(f)
                escritor.writerow([nombre, fecha, hora])

    cv2.imshow("Asistencia Automatica - Colegio", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

	A
1	Nombre,Fecha,Hora
2	EDGAR,2025-08-26,11:14:31
3	EDGAR,2025-08-26,11:14:31
4	EDGAR,2025-08-26,11:14:32
5	EDGAR,2025-08-26,11:14:32
6	EDGAR,2025-08-26,11:14:32
7	EDGAR,2025-08-26,11:14:33
8	EDGAR,2025-08-26,11:14:33