¡Entiendo! Si no tienes un modelo preentrenado para imágenes, podemos crear uno desde cero usando \*\*Keras\*\* y \*\*TensorFlow\*\*, y entrenarlo con un conjunto de datos de imágenes de señales de "Stop" y "No Stop". A continuación te guiaré en la creación del modelo y en el entrenamiento con tus propias imágenes.

### 1. \*\*Preparación de los datos\*\*

Lo primero que necesitas es un conjunto de imágenes organizadas en carpetas. La estructura de directorios podría ser la siguiente:

```

dataset/

├── train/

│ ├── stop/

│ │ ├── stop1.jpg

│ │ ├── stop2.jpg

│ └── not\_stop/

│ ├── not\_stop1.jpg

│ ├── not\_stop2.jpg

├── validation/

│ ├── stop/

│ │ ├── stop\_val1.jpg

│ └── not\_stop/

│ ├── not\_stop\_val1.jpg

```

### 2. \*\*Creación del modelo de clasificación de imágenes\*\*

A continuación te proporcionaré un código para construir y entrenar un modelo de \*\*red neuronal convolucional (CNN)\*\* para clasificar imágenes de señales de tráfico ("Stop" y "No Stop").

#### Código para entrenar un modelo CNN:

```python

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Configuración de generadores de datos con aumentación para evitar sobreajuste

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest',

validation\_split=0.2 # 20% de los datos para validación

)

# Cargar imágenes desde la carpeta

train\_generator = train\_datagen.flow\_from\_directory(

'dataset/train', # Ruta donde están las imágenes de entrenamiento

target\_size=(224, 224), # Tamaño de las imágenes (ajústalo según tu modelo)

batch\_size=32,

class\_mode='binary',

subset='training' # Subconjunto de entrenamiento

)

validation\_generator = train\_datagen.flow\_from\_directory(

'dataset/train', # Usamos la misma ruta, pero separamos la validación

target\_size=(224, 224),

batch\_size=32,

class\_mode='binary',

subset='validation' # Subconjunto de validación

)

# Definir el modelo CNN

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D(pool\_size=(2, 2)),

Flatten(),

Dense(128, activation='relu'),

Dense(1, activation='sigmoid') # Una salida binaria (Stop/No Stop)

])

# Compilar el modelo

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Entrenar el modelo

history = model.fit(

train\_generator,

validation\_data=validation\_generator,

epochs=10 # Puedes aumentar el número de épocas si es necesario

)

# Guardar el modelo entrenado

model.save('stop\_sign\_classifier.h5')

# Evaluar el modelo en los datos de validación

val\_loss, val\_accuracy = model.evaluate(validation\_generator)

print(f"Loss en validación: {val\_loss}")

print(f"Accuracy en validación: {val\_accuracy}")

```

### 3. \*\*Predecir en nuevas imágenes\*\*

Después de entrenar el modelo, puedes usarlo para realizar predicciones en nuevas imágenes de señales de tráfico. Aquí te dejo un código para predecir en imágenes y mostrar los resultados.

#### Código para predecir en nuevas imágenes:

```python

import cv2

import numpy as np

from tensorflow.keras.models import load\_model

import matplotlib.pyplot as plt

# Cargar el modelo entrenado

model = load\_model('stop\_sign\_classifier.h5')

# Función para predecir en una imagen

def predict\_image(image\_path):

# Cargar la imagen

image = cv2.imread(image\_path)

# Redimensionar la imagen

image\_resized = cv2.resize(image, (224, 224))

# Normalizar los valores de la imagen

image\_normalized = image\_resized / 255.0

# Expandir dimensiones para que coincida con la entrada del modelo

image\_array = np.expand\_dims(image\_normalized, axis=0)

# Realizar la predicción

prediction = model.predict(image\_array)

# Mostrar la imagen y la predicción

plt.imshow(cv2.cvtColor(image\_resized, cv2.COLOR\_BGR2RGB))

plt.title(f"Prediction = {'Stop' if prediction[0] > 0.5 else 'Not Stop'}")

plt.show()

# Probar con una imagen nueva

predict\_image('path\_to\_test\_image.jpeg') # Ruta de la imagen de prueba

```

### 4. \*\*Documentación del Proyecto\*\*

Después de entrenar y realizar predicciones, sigue los pasos solicitados por la actividad:

1. \*\*Captura de Pantallas\*\*: Toma capturas de pantalla de las predicciones en las imágenes de prueba, tanto para las señales de "Stop" como para las imágenes sin señales de "Stop".

2. \*\*Entrenamiento Adicional\*\*: Si alguna imagen es mal clasificada, añade más datos relevantes para mejorar la precisión del modelo. Retrénalo con los nuevos datos y muestra cómo el modelo mejora.

3. \*\*Subir los resultados\*\*: Presenta las capturas de pantalla y las mejoras implementadas al modelo, siguiendo las instrucciones de la actividad en \*\*Coursera\*\*.

### Resumen de pasos:

1. \*\*Preparar las imágenes\*\* para entrenar y validar el modelo.

2. \*\*Entrenar un modelo CNN\*\* con Keras usando las imágenes de "Stop" y "No Stop".

3. \*\*Hacer predicciones\*\* en nuevas imágenes y mostrar los resultados.

4. \*\*Reentrenar el modelo\*\* si hay errores de clasificación, añadiendo más datos relevantes.

Si tienes alguna duda o necesitas más detalles en alguna parte del código, ¡no dudes en preguntarme! Estoy aquí para ayudarte a completar el proyecto.

Aquí te detallo, en orden, qué puedes ir subiendo a la actividad de Coursera según las capturas de pantalla que hayas obtenido:

### 1. \*\*Upload the screenshots of your results from the 1st image for a stop sign\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*primera imagen\*\* de una señal de \*\*Stop\*\* que hayas usado.

### 2. \*\*Upload the screenshots of your results from the 2nd image for a traffic stop sign\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*segunda imagen\*\* de una señal de \*\*Stop\*\*.

### 3. \*\*Upload the screenshots of your results from the 3rd image for a traffic stop sign\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*tercera imagen\*\* de una señal de \*\*Stop\*\*.

### 4. \*\*Upload the screenshots of your results from the 4th image for a traffic stop sign\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*cuarta imagen\*\* de una señal de \*\*Stop\*\*.

### 5. \*\*Upload the screenshots of your results from the 1st image for a "not -stop sign"\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*primera imagen\*\* de una señal de \*\*No Stop\*\*.

### 6. \*\*Upload the screenshots of your results from the 2nd image for a "not -stop sign"\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*segunda imagen\*\* de una señal de \*\*No Stop\*\*.

### 7. \*\*Upload the screenshots of your results from the 3rd image for a "not -stop sign"\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*tercera imagen\*\* de una señal de \*\*No Stop\*\*.

### 8. \*\*Upload the screenshots of your results from the final image for a "not -stop sign"\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para la \*\*cuarta imagen\*\* de una señal de \*\*No Stop\*\*.

### 9. \*\*Find an image of a stop sign online and classify it correctly\*\*

- Aquí deberás subir la \*\*captura de pantalla\*\* de la predicción correcta para una señal de \*\*Stop\*\* que hayas encontrado y descargado de \*\*internet\*\* (como la sugerida anteriormente).

### 10. \*\*Find an image of a street without a stop sign online and classify it correctly\*\*

- Subir la \*\*captura de pantalla\*\* de la predicción correcta para una imagen de una \*\*calle sin una señal de Stop\*\*, descargada de internet.

### 11. \*\*Find an image of a street without a stop sign or with a stop sign that will be incorrectly classified by your classifier\*\*

- Subir la \*\*captura de pantalla\*\* de una imagen que haya sido \*\*incorrectamente clasificada\*\* por el modelo (puede ser una señal confusa o una calle sin señal que se clasifica erróneamente).

### 12. \*\*Find more data and re-train your classifier to classify the misclassified sample, show your new classifier works on the misclassified image\*\*

- Una vez que tengas una \*\*muestra mal clasificada\*\* (como la del paso anterior), añade más datos para corregir ese tipo de error, reentrena el modelo, y sube la \*\*captura de pantalla\*\* mostrando que el \*\*modelo reentrenado\*\* clasifica correctamente la imagen que antes estaba mal.

Con estas instrucciones, puedes ir organizando tus capturas de pantalla y subiendo los resultados a la actividad. ¡Hazme saber si necesitas ayuda adicional!