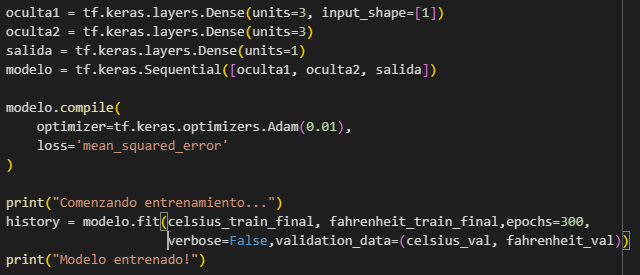
Javier Díaz Machado

Tensorflow.JS

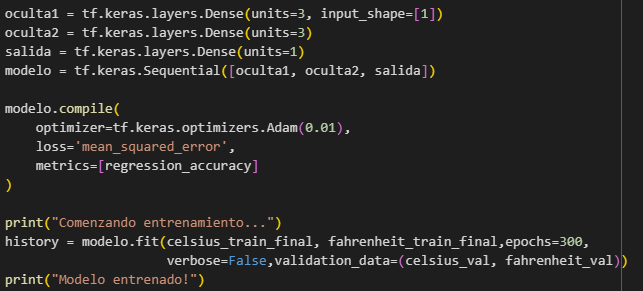
[Subtítulo del documento]

**1** (4 Puntos) Realiza la tarea de implementar un modelo para convertir temperaturas de grados Fahrenheit a centígrados. Expórtalo a Tensorflow.js e implementa la aplicación web para que use el modelo.

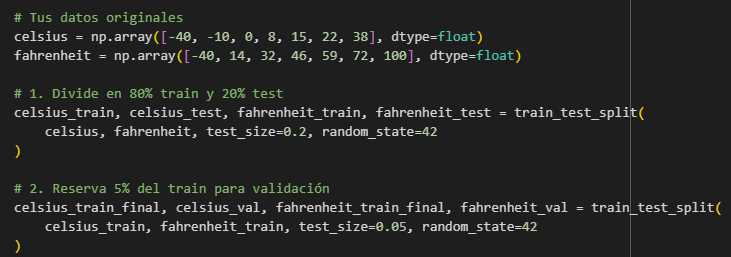
Compilamos y entrenamos el modelo:



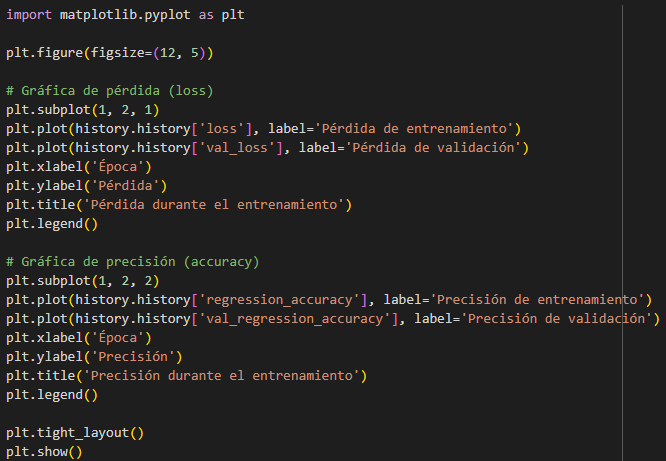
Descarga la función de conversión y genera el dataset .csv con al menos 1000 temperaturas.

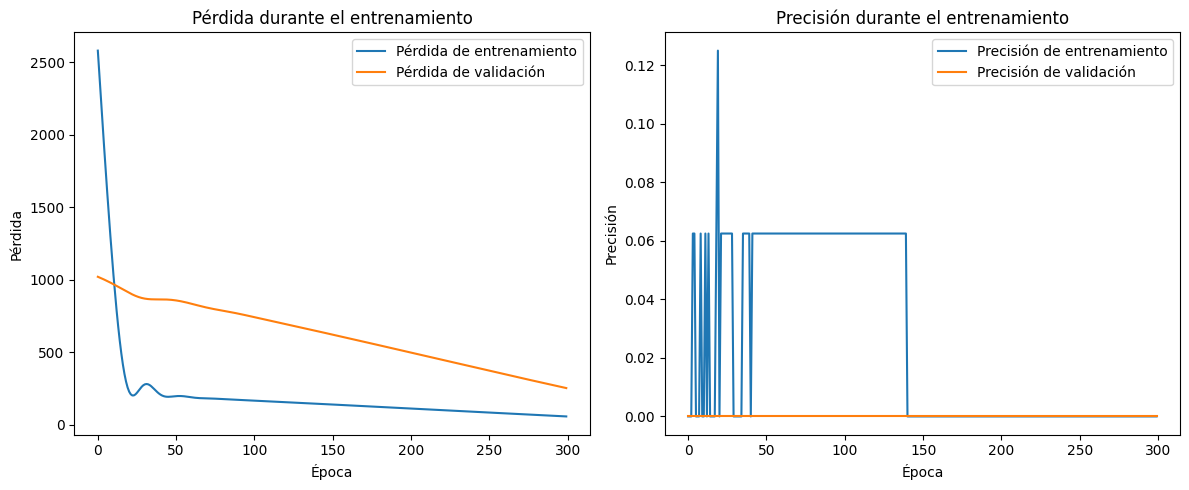


Divide los datos en 80% training y 20% test. Los datos de trainning reserva un 5% para validación.



Muestra las gráficas de pérdida y precisión.





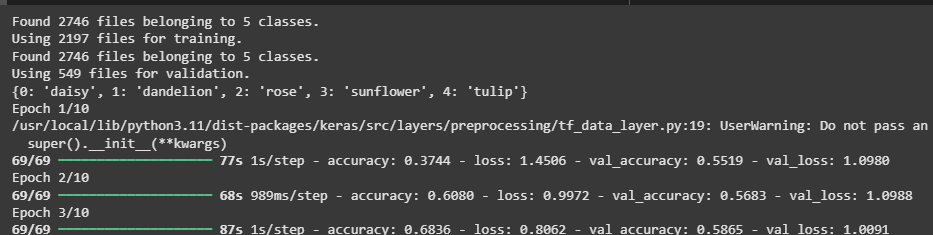
**2** (6 Puntos) Descarga el data set de flores de:

https://www.kaggle.com/datasets/imsparsh/flowers-dataset?resource=download



Implementa en Python el modelo de red convolucional que clasifique correctamente las flores, expórtalo y úsalo en una aplicación web en la que se seleccionará una imagen e indicará su nombre.

|  |
| --- |
| import tensorflow as tf  IMG\_WIDTH = 180  IMG\_HEIGHT = 180  BATCH\_SIZE = 32  # Cargar datasets desde la carpeta 'dataset/'  train\_ds = tf.keras.utils.image\_dataset\_from\_directory(      "dataset",      validation\_split=0.2,      subset="training",      seed=123,      image\_size=(IMG\_HEIGHT, IMG\_WIDTH),      batch\_size=BATCH\_SIZE  )  val\_ds = tf.keras.utils.image\_dataset\_from\_directory(      "dataset",      validation\_split=0.2,      subset="validation",      seed=123,      image\_size=(IMG\_HEIGHT, IMG\_WIDTH),      batch\_size=BATCH\_SIZE  )  # Obtener nombres de las clases  class\_names = train\_ds.class\_names  print(class\_names)  import tensorflow as tf  import json  IMG\_WIDTH = 180  IMG\_HEIGHT = 180  BATCH\_SIZE = 32  # Cargar datasets desde la carpeta 'dataset/train/'  train\_ds = tf.keras.utils.image\_dataset\_from\_directory(      "dataset/train",      validation\_split=0.2,      subset="training",      seed=123,      image\_size=(IMG\_HEIGHT, IMG\_WIDTH),      batch\_size=BATCH\_SIZE  )  val\_ds = tf.keras.utils.image\_dataset\_from\_directory(      "dataset/train",      validation\_split=0.2,      subset="validation",      seed=123,      image\_size=(IMG\_HEIGHT, IMG\_WIDTH),      batch\_size=BATCH\_SIZE  )  # Obtener nombres de las clases  class\_names = {i: name for i, name in enumerate(train\_ds.class\_names)}  print(class\_names)    # Configurar datasets para rendimiento  AUTOTUNE = tf.data.AUTOTUNE  train\_ds = train\_ds.cache().shuffle(1000).prefetch(buffer\_size=AUTOTUNE)  val\_ds = val\_ds.cache().prefetch(buffer\_size=AUTOTUNE)  # Definir el modelo CNN  num\_classes = len(class\_names)  model = tf.keras.Sequential([      tf.keras.layers.Rescaling(1./255, input\_shape=(IMG\_HEIGHT, IMG\_WIDTH, 3)),      tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu'),      tf.keras.layers.MaxPooling2D(),      tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),      tf.keras.layers.MaxPooling2D(),      tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),      tf.keras.layers.MaxPooling2D(),      tf.keras.layers.Flatten(),      tf.keras.layers.Dense(128, activation='relu'),      tf.keras.layers.Dense(num\_classes)  ])  # Compilar el modelo  model.compile(      optimizer='adam',      loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),      metrics=['accuracy']  )  # Entrenar el modelo  epochs = 10  history = model.fit(      train\_ds,      validation\_data=val\_ds,      epochs=epochs  )  # Exportar el modelo  model.save('flower\_classifier.h5')  # Convertir el modelo a formato TensorFlow.js  import subprocess  subprocess.run(["mkdir", "flower\_classifier\_tfjs"])  subprocess.run([      "tensorflowjs\_converter",      "--input\_format", "keras",      "flower\_classifier.h5",      "flower\_classifier\_tfjs"  ])  with open("flower\_classifier\_tfjs/class\_names.json", "w") as f:      json.dump(class\_names, f) |



Intenté levanter el servidor en flask, pero al cambiar de colab a vscode me dio problemas.