Inteligencia artificial avanzada para la ciencia de datos II

(Gpo 501)

Momento de Retroalimentación Individual: Implementación de un modelo de Deep Learning

Alumna:

Ana Lucía Cárdenas Pérez A01284090

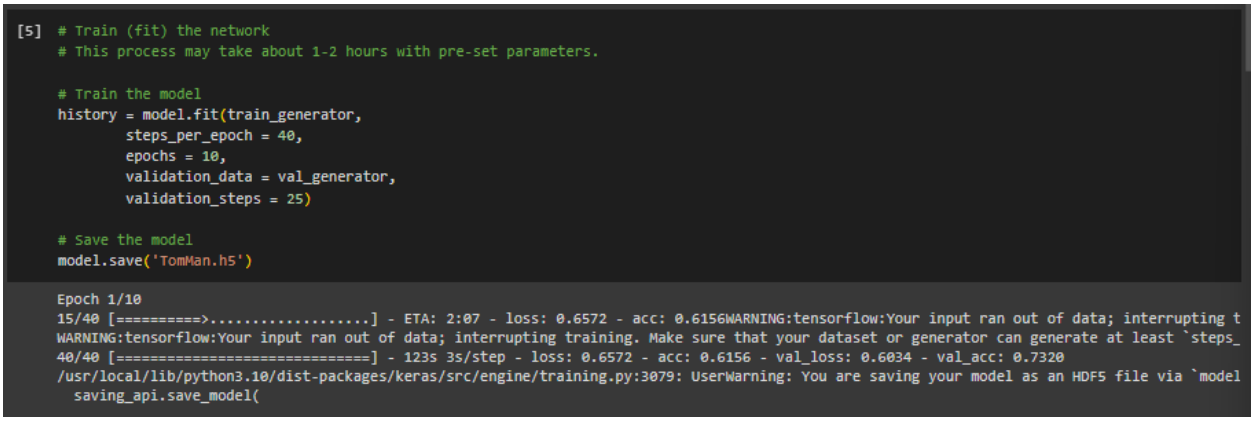
Profesor

Christian Carlos Mendoza Buenrostro

Monterrey, Nuevo León a 4 de noviembre del 2023

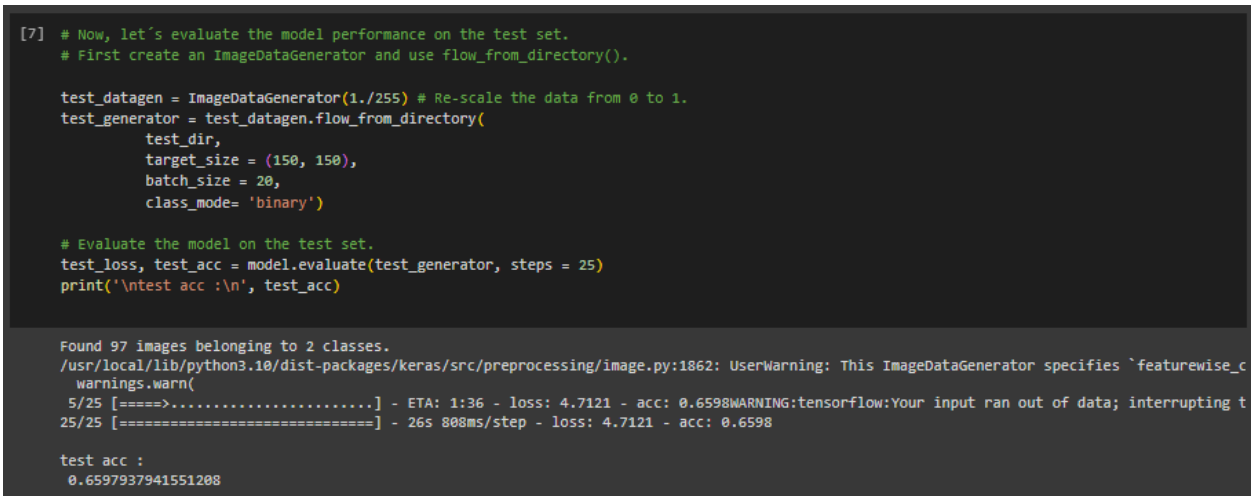
Para esta entrega individual decidí trabajar con uno de los modelos vistos en clase el cual trabaja con Tensorflow y Keras. Este código se usó para entrenar un modelo de clasificación binaria de imágenes, se utilizó VGG16 el cual es un modelo pre-entrenado con los miles de datos que se encuentran en ImageNet. Con esto se busca poder clasificar imágenes en 2 categorías. En este caso se trabajó con un set de datos encontrado en Kaggle el cuál contiene imágenes de manzanas y tomates. Decidí utilizar este set de datos ya que a pesar de que sea un set de datos sencillo, estas dos frutas son similares físicamente en las imágenes por su forma y su color.

Se llevaron a cabo 3 pruebas para obtener el mejor resultado posible.
Para la primera implementación los valores de entrenamiento fueron, para “steps_per_epoch = 40”, “epochs = 10”, y “validation_steps = 25”. Y al evaluar el modelo se usaron los valores de “batch_size = 20” en el test_generator. Al final probamos el accuracy y se obtuvo un valor de 0.6597. Aunque es un buen resultado, lo que buscamos con este data set es que este sea lo más accurate posible por la similaridad en las dos categorías.



```
[5]  # Train (fit) the network
     # This process may take about 1-2 hours with pre-set parameters.

     # Train the model
     history = model.fit(train_generator,
             steps_per_epoch = 40,
             epochs = 10,
             validation_data = val_generator,
             validation_steps = 25)

     # Save the model
     model.save('TomMan.h5')
```

```
Epoch 1/10
15/40 [==========>...................] - ETA: 2:07 - loss: 0.6572 - acc: 0.6156WARNING:tensorflow:Your input ran out of data; interrupting t
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_
40/40 [==============================] - 123s 3s/step - loss: 0.6572 - acc: 0.6156 - val_loss: 0.6034 - val_acc: 0.7320
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model
  saving_api.save_model(
```



```
[7]  # Now, let´s evaluate the model performance on the test set.
     # First create an ImageDataGenerator and use flow_from_directory().

     test_datagen = ImageDataGenerator(1./255) # Re-scale the data from 0 to 1.
     test_generator = test_datagen.flow_from_directory(
              test_dir,
              target_size = (150, 150),
              batch_size = 20,
              class_mode= 'binary')

     # Evaluate the model on the test set.
     test_loss, test_acc = model.evaluate(test_generator, steps = 25)
     print('\ntest acc :\n', test_acc)
```

```
Found 97 images belonging to 2 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1862: UserWarning: This ImageDataGenerator specifies `featurewise_c
  warnings.warn(
 5/25 [=====>........................] - ETA: 1:36 - loss: 4.7121 - acc: 0.6598WARNING:tensorflow:Your input ran out of data; interrupting t
25/25 [==============================] - 26s 808ms/step - loss: 4.7121 - acc: 0.6598

test acc :
 0.6597937941551208
```

Para la segunda implementación los valores de entrenamiento fueron, para “steps_per_epoch = 50”, “epochs = 10”, y “validation_steps = 35”. Y al evaluar el modelo se usó el valor de “batch_size = 20” en el test_generator. Al final probamos el accuracy del modelo y se obtuvo un valor de 0.5876. Los cambios de los valores de la segunda prueba nos dieron resultados más bajos, por lo que no sería un conjunto bueno para la predicción.

```
[56] # Train (fit) the network
     # This process may take about 1-2 hours with pre-set parameters.

     # Train the model
     history = model.fit(train_generator,
             steps_per_epoch = 50,
             epochs = 10,
             validation_data = val_generator,
             validation_steps = 35)

     # Save the model
     model.save('TomMan.h5')
```

```
Epoch 1/10
15/50 [========>.....................] - ETA: 2:57 - loss: 0.6703 - acc: 0.6259WARNING:tensorflow:Your input ran out of data; interrupting t
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_
50/50 [==============================] - 121s 2s/step - loss: 0.6703 - acc: 0.6259 - val_loss: 0.6005 - val_acc: 0.6804
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model
  saving_api.save_model(
```

```
[58] # Now, let´s evaluate the model performance on the test set.
     # First create an ImageDataGenerator and use flow_from_directory().

     test_datagen = ImageDataGenerator(1./255) # Re-scale the data from 0 to 1.
     test_generator = test_datagen.flow_from_directory(
              test_dir,
              target_size = (150, 150),
              batch_size = 20,
              class_mode= 'binary')

     # Evaluate the model on the test set.
     test_loss, test_acc = model.evaluate(test_generator, steps = 25)
     print('\ntest acc :\n', test_acc)
```

```
Found 97 images belonging to 2 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1862: UserWarning: This ImageDataGenerator specifies `featurewise_c
  warnings.warn(
 5/25 [=====>........................] - ETA: 1:35 - loss: 5.7139 - acc: 0.5876WARNING:tensorflow:Your input ran out of data; interrupting t
25/25 [==============================] - 26s 797ms/step - loss: 5.7139 - acc: 0.5876

test acc :
 0.5876288414001465
```

Para la última implementación los valores de entrenamiento fueron para “steps_per_epoch = 40”, “epochs = 10”, y “validation_steps = 35”. Para evaluar el modelo se utilizó “batch_size = 20” en el test_generator. Al final también probamos el accuracy y se obtuvo un valor de 0.7319. Este fue el mejor resultado que obtuvimos y con el que llevamos a cabo la predicción con las imágenes de prueba.

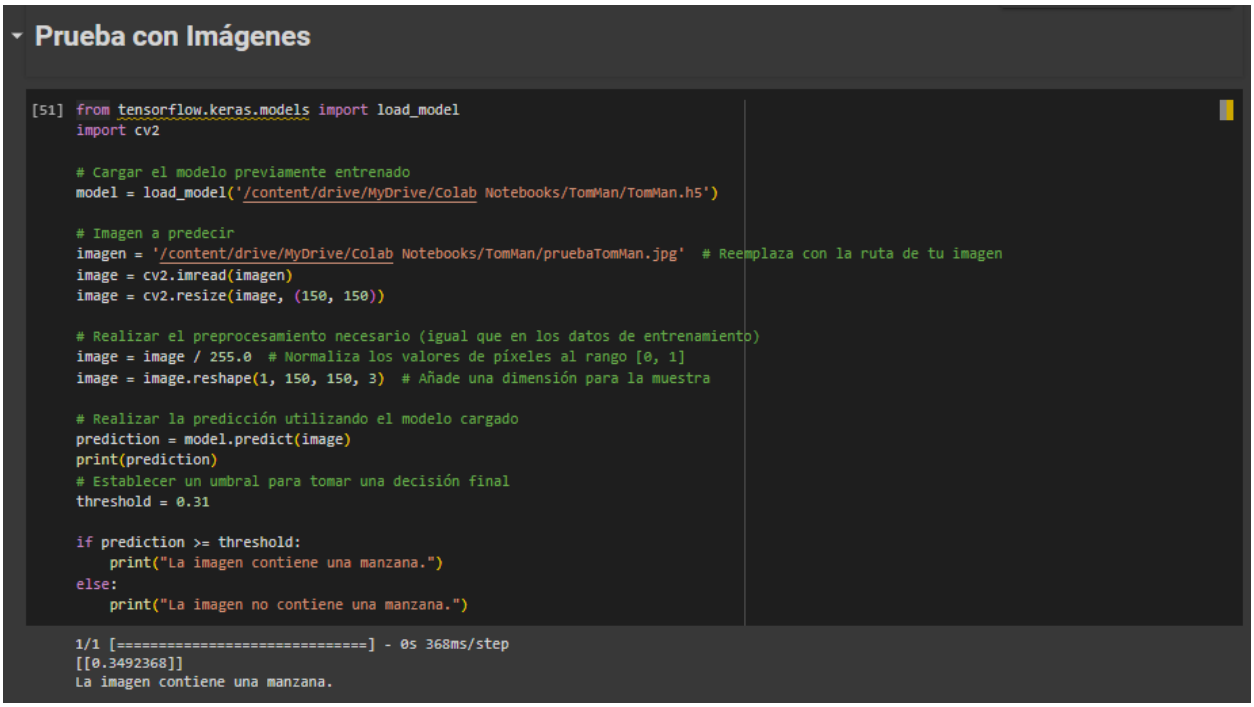
```
# Train (fit) the network
# This process may take about 1-2 hours with pre-set parameters.

# Train the model
history = model.fit(train_generator,
        steps_per_epoch = 40,
        epochs = 10,
        validation_data = val_generator,
        validation_steps = 35)

# Save the model
model.save('TomMan.h5')
```

```
Epoch 1/10
15/40 [==========>...................] - ETA: 2:10 - loss: 0.5918 - acc: 0.7347WARNING:tensorflow:Your input ran out of data; interrupting t
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_
40/40 [==============================] - 105s 3s/step - loss: 0.5918 - acc: 0.7347 - val_loss: 0.5465 - val_acc: 0.7629
```

```
# Now, let´s evaluate the model performance on the test set.
# First create an ImageDataGenerator and use flow_from_directory().

test_datagen = ImageDataGenerator(1./255) # Re-scale the data from 0 to 1.
test_generator = test_datagen.flow_from_directory(
          test_dir,
          target_size = (150, 150),
          batch_size = 20,
          class_mode= 'binary')

# Evaluate the model on the test set.
test_loss, test_acc = model.evaluate(test_generator, steps = 25)
print('\ntest acc :\n', test_acc)
```

```
Found 97 images belonging to 2 classes.
 5/25 [=====>........................] - ETA: 1:45 - loss: 4.9928 - acc: 0.7320WARNING:tensorflow:Your input ran out of data; interrupting t
25/25 [==============================] - 26s 876ms/step - loss: 4.9928 - acc: 0.7320

test acc :
 0.7319587469100952
```

Ahora para la prueba con imágenes, escogimos el modelo con mejor resultado en accuracy y cargamos la imagen que queremos utilizar para generar una predicción. El modelo utilizado nos da una función para obtener el valor de predicción, haciendo algunas pruebas con diferentes fotos, pudimos encontrar un valor de threshold con el que vamos a poder determinar si una imagen es una manzana o si es un tomate, este valor fue de aproximadamente 0.31.



Referencias:

Cargar una red neuronal convolucional VGG-16 preentrenada. (2023). Retrieved November 5, 2023, from Mathworks.com website: https://la.mathworks.com/help/deeplearning/ref/vgg16.html

Cortinhas, S. (2022). Apples or tomatoes - image classification. Retrieved November 5, 2023, from Kaggle.com website: https://www.kaggle.com/datasets/samuelcortinhas/apples-or-tomatoes-image-classification