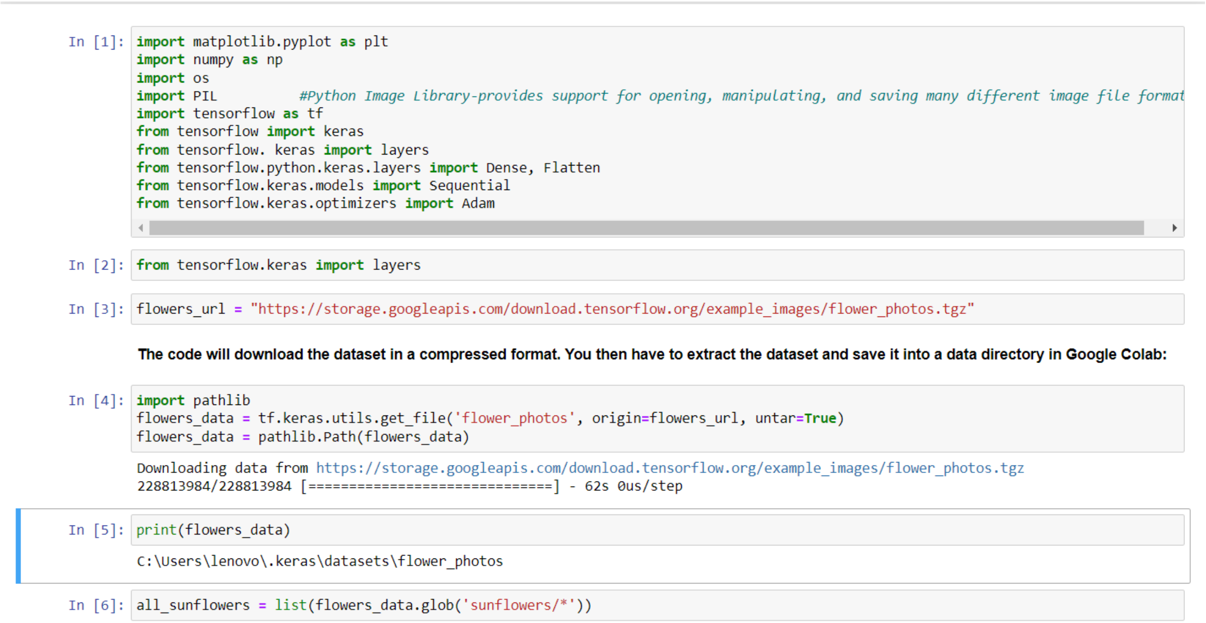


```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import os
        import PIL          #Python Image Library-provides support for opening, manipulating, and saving many different image file format
        import tensorflow as tf
        from tensorflow import keras
        from tensorflow. keras import layers
        from tensorflow.python.keras.layers import Dense, Flatten
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.optimizers import Adam
```

```
In [2]: from tensorflow.keras import layers
```

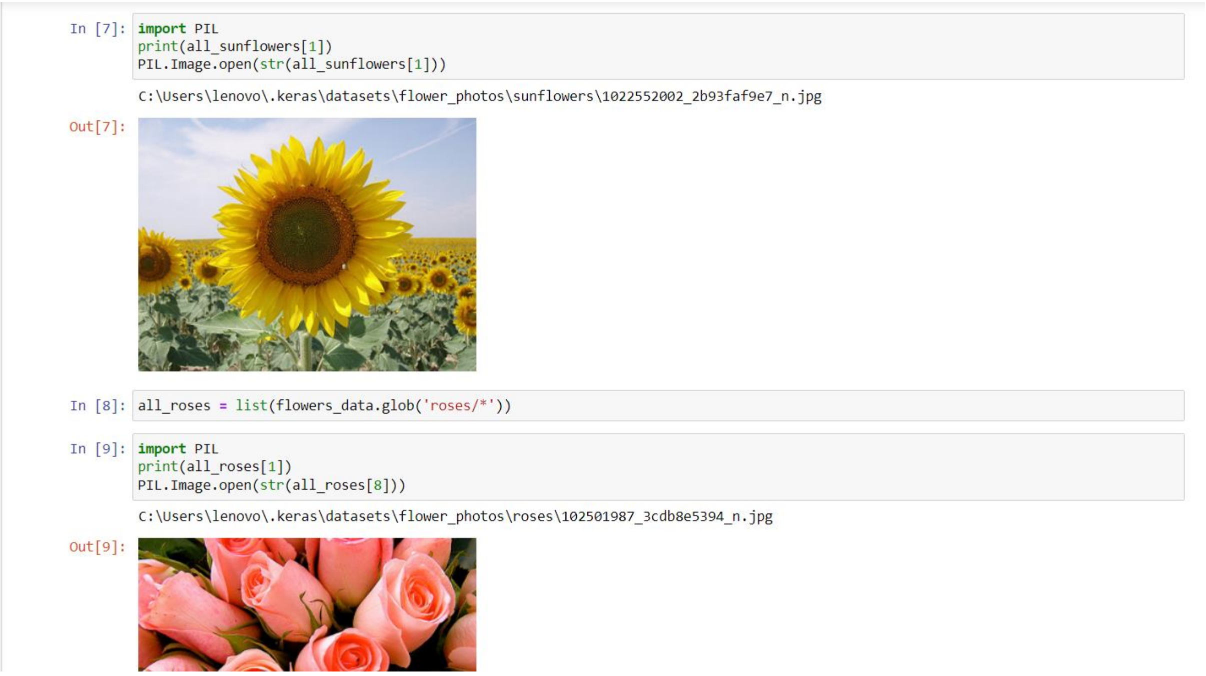
```
In [3]: flowers_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
```

The code will download the dataset in a compressed format. You then have to extract the dataset and save it into a data directory in Google Colab:

```
In [4]: import pathlib
        flowers_data = tf.keras.utils.get_file('flower_photos', origin=flowers_url, untar=True)
        flowers_data = pathlib.Path(flowers_data)

        Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
        228813984/228813984 [==============================] - 62s 0us/step
```

```
In [5]: print(flowers_data)

        C:\Users\lenovo\.keras\datasets\flower_photos
```

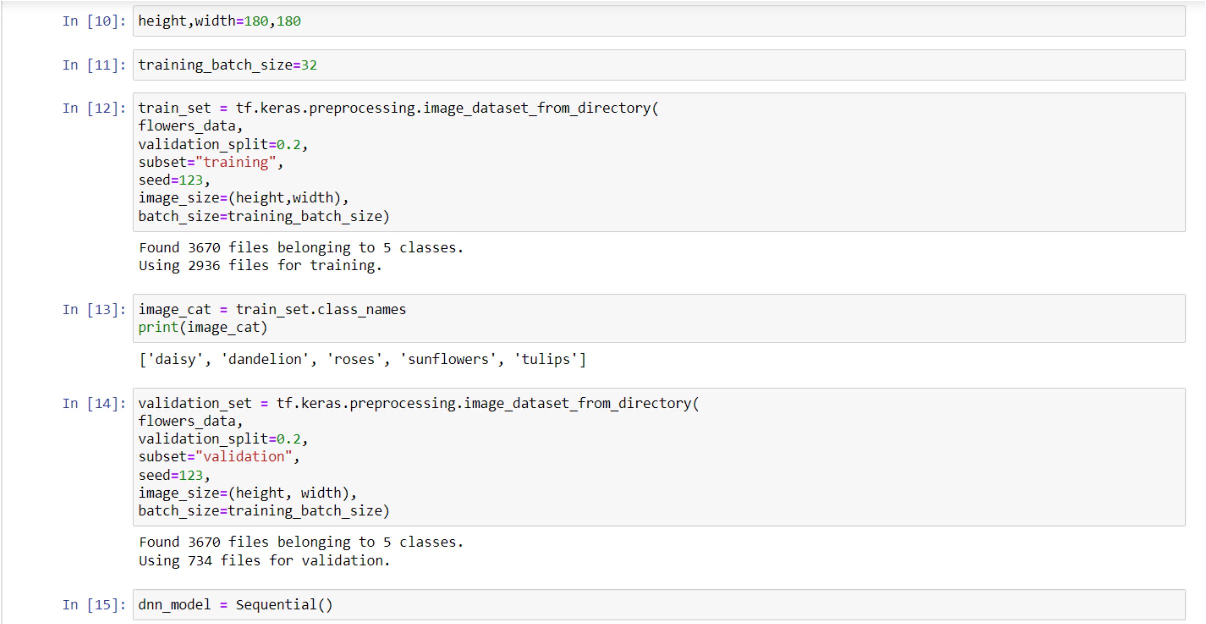
```
In [6]: all_sunflowers = list(flowers_data.glob('sunflowers/*'))
```



```
In [7]: import PIL
        print(all_sunflowers[1])
        PIL.Image.open(str(all_sunflowers[1]))

        C:\Users\lenovo\.keras\datasets\flower_photos\sunflowers\1022552002_2b93faf9e7_n.jpg

Out[7]:
```

```
In [8]: all_roses = list(flowers_data.glob('roses/*'))
```

```
In [9]: import PIL
        print(all_roses[1])
        PIL.Image.open(str(all_roses[8]))

        C:\Users\lenovo\.keras\datasets\flower_photos\roses\102501987_3cdb8e5394_n.jpg

Out[9]:
```



```
In [10]: height,width=180,180
```

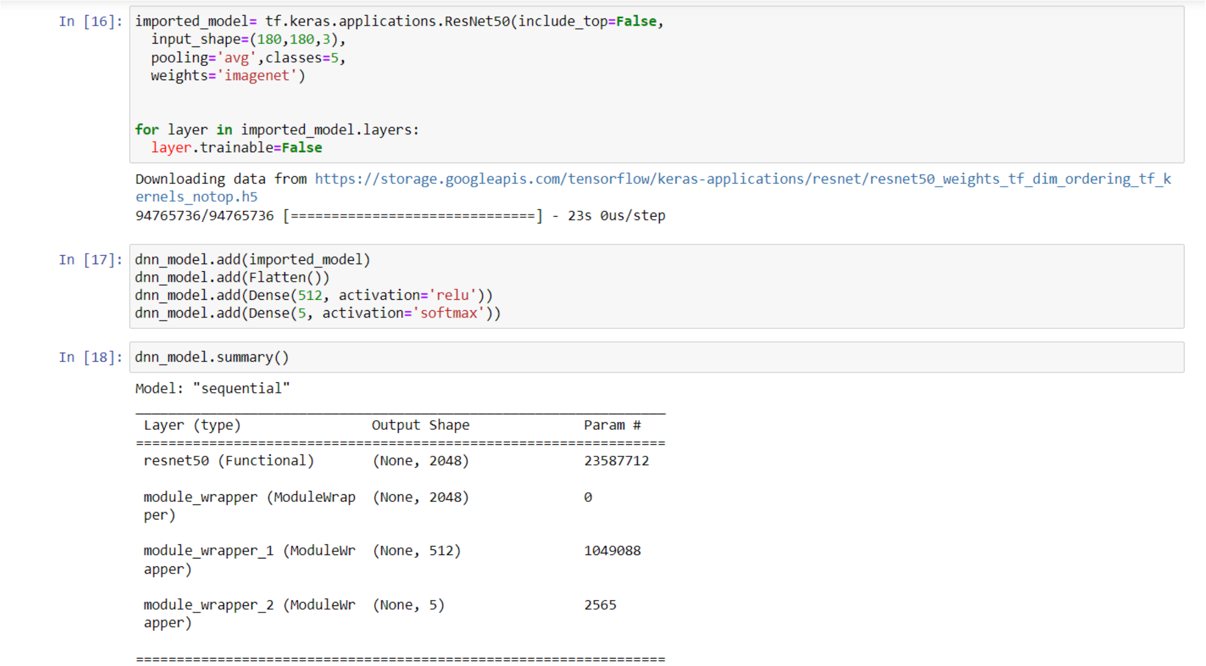
```
In [11]: training_batch_size=32
```

```
In [12]: train_set = tf.keras.preprocessing.image_dataset_from_directory(
         flowers_data,
         validation_split=0.2,
         subset="training",
         seed=123,
         image_size=(height,width),
         batch_size=training_batch_size)

         Found 3670 files belonging to 5 classes.
         Using 2936 files for training.
```

```
In [13]: image_cat = train_set.class_names
         print(image_cat)

         ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

```
In [14]: validation_set = tf.keras.preprocessing.image_dataset_from_directory(
         flowers_data,
         validation_split=0.2,
         subset="validation",
         seed=123,
         image_size=(height, width),
         batch_size=training_batch_size)

         Found 3670 files belonging to 5 classes.
         Using 734 files for validation.
```

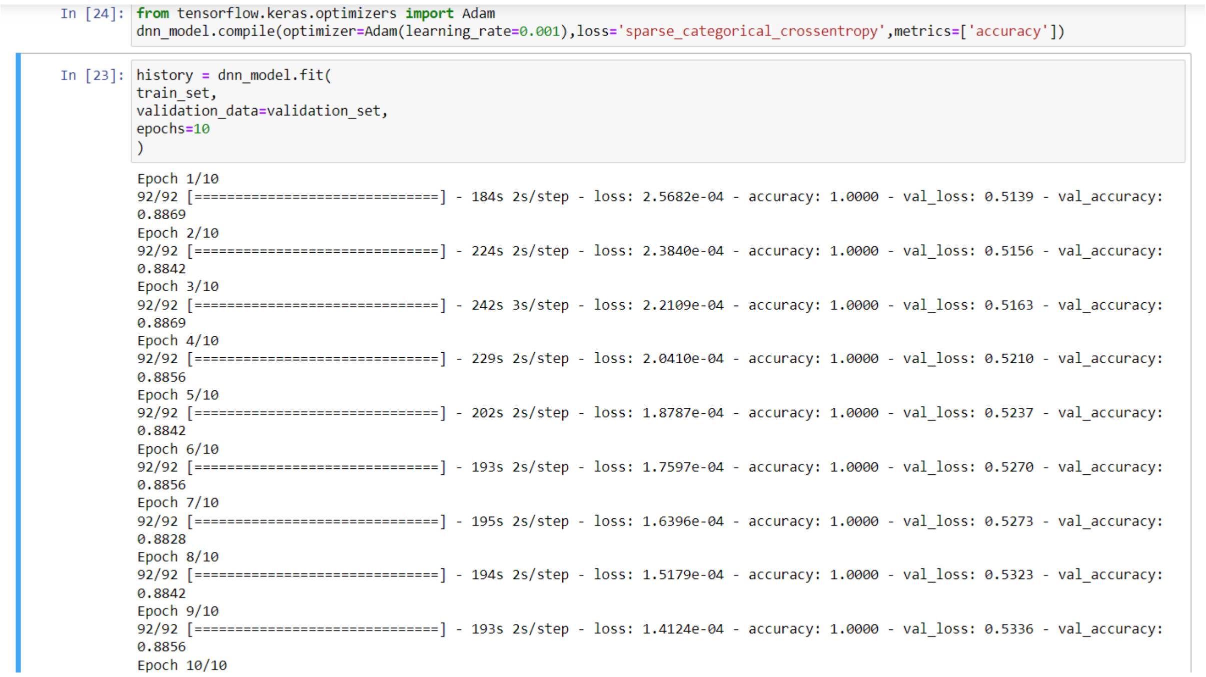
```
In [15]: dnn_model = Sequential()
```



```
In [16]: imported_model= tf.keras.applications.ResNet50(include_top=False,
           input_shape=(180,180,3),
           pooling='avg',classes=5,
           weights='imagenet')


         for layer in imported_model.layers:
           layer.trainable=False

         Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_k
         ernels_notop.h5
         94765736/94765736 [==============================] - 23s 0us/step
```

```
In [17]: dnn_model.add(imported_model)
         dnn_model.add(Flatten())
         dnn_model.add(Dense(512, activation='relu'))
         dnn_model.add(Dense(5, activation='softmax'))
```

```
In [18]: dnn_model.summary()

         Model: "sequential"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          resnet50 (Functional)       (None, 2048)              23587712

          module_wrapper (ModuleWrap  (None, 2048)              0
          per)

          module_wrapper_1 (ModuleWr  (None, 512)               1049088
          apper)

          module_wrapper_2 (ModuleWr  (None, 5)                 2565
          apper)

         =================================================================
```



In [24]:

```
from tensorflow.keras.optimizers import Adam
dnn_model.compile(optimizer=Adam(learning_rate=0.001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

In [23]:

```
history = dnn_model.fit(
train_set,
validation_data=validation_set,
epochs=10
)
```

```
Epoch 1/10
92/92 [==============================] - 184s 2s/step - loss: 2.5682e-04 - accuracy: 1.0000 - val_loss: 0.5139 - val_accuracy:
0.8869
Epoch 2/10
92/92 [==============================] - 224s 2s/step - loss: 2.3840e-04 - accuracy: 1.0000 - val_loss: 0.5156 - val_accuracy:
0.8842
Epoch 3/10
92/92 [==============================] - 242s 3s/step - loss: 2.2109e-04 - accuracy: 1.0000 - val_loss: 0.5163 - val_accuracy:
0.8869
Epoch 4/10
92/92 [==============================] - 229s 2s/step - loss: 2.0410e-04 - accuracy: 1.0000 - val_loss: 0.5210 - val_accuracy:
0.8856
Epoch 5/10
92/92 [==============================] - 202s 2s/step - loss: 1.8787e-04 - accuracy: 1.0000 - val_loss: 0.5237 - val_accuracy:
0.8842
Epoch 6/10
92/92 [==============================] - 193s 2s/step - loss: 1.7597e-04 - accuracy: 1.0000 - val_loss: 0.5270 - val_accuracy:
0.8856
Epoch 7/10
92/92 [==============================] - 195s 2s/step - loss: 1.6396e-04 - accuracy: 1.0000 - val_loss: 0.5273 - val_accuracy:
0.8828
Epoch 8/10
92/92 [==============================] - 194s 2s/step - loss: 1.5179e-04 - accuracy: 1.0000 - val_loss: 0.5323 - val_accuracy:
0.8842
Epoch 9/10
92/92 [==============================] - 193s 2s/step - loss: 1.4124e-04 - accuracy: 1.0000 - val_loss: 0.5336 - val_accuracy:
0.8856
Epoch 10/10
```