9_5_tf_deployment_Flask

September 21, 2021

1 Deploy a TensorFlow model with Flask

1.0.1 Problem Statement

Deploy a neural network model to classify images of clothing, like sneakers and shirts

1.0.2 The Fashion MNIST dataset overview

The Fashion MNIST dataset contains 70,000 grayscale images in 10 categories. The images show individual articles of clothing at low resolution (28 by 28 pixels), as seen below:

```
<img src="https://tensorflow.org/images/fashion-mnist-sprite.png"
     alt="Fashion MNIST sprite"  width="600">
```

```
<b>Figure 1.</b> <a href="https://github.com/zalandoresearch/fashion-mnist">Fashion-MNIST samp
```

Although these are really images, they are loaded as NumPy arrays and not as binary image objects.

1.0.3 Import the libraries

[ ]:
```
import warnings
warnings.filterwarnings("ignore")
```

[ ]:
```
import keras
import tensorflow as tf
```

```
Using TensorFlow backend.
```

1.0.4 Load the dataset

[ ]:
```
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.
 ↪load_data()
```

[ ]:
```
# scale the values to 0.0 to 1.0
train_images = train_images / 255.0
```

```
test_images = test_images / 255.0
```

[ ]:
```
# reshape for feeding into the model
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1)
test_images = test_images.reshape(test_images.shape[0], 28, 28, 1)
```

[ ]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

1.0.5 Define the model

[ ]:
```
model = keras.Sequential([
  keras.layers.Conv2D(input_shape=(28,28,1), filters=8, kernel_size=3,
                      strides=2, activation='relu', name='Conv1'),
  keras.layers.Flatten(),
  keras.layers.Dense(10, activation='softmax',name='Softmax')
])
```

1.0.6 Train the model

[ ]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[ ]:
```
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 6s 103us/step - loss: 0.5440 -
accuracy: 0.8138
Epoch 2/5
60000/60000 [==============================] - 6s 102us/step - loss: 0.3854 -
accuracy: 0.8647
Epoch 3/5
60000/60000 [==============================] - 6s 108us/step - loss: 0.3474 -
accuracy: 0.8788
Epoch 4/5
60000/60000 [==============================] - 6s 103us/step - loss: 0.3260 -
accuracy: 0.8841
Epoch 5/5
60000/60000 [==============================] - 6s 104us/step - loss: 0.3116 -
accuracy: 0.8898
```

[ ]:
```
<keras.callbacks.callbacks.History at 0x7f00034e9b00>
```

1.0.7 Evaluate the model

[ ]:
```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('\nTest accuracy: {}'.format(test_acc))
```

```
10000/10000 [==============================] - 1s 53us/step

Test accuracy: 0.8791000247001648
```

1.0.8 Save the model

[ ]:
```
model.save("model.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

1.0.9 Serve the model with Flask

[ ]:
```
# Load libraries
import flask
from keras.models import load_model
from flask import request
import numpy as np
```

[ ]:
```
# instantiate flask
app = flask.Flask(__name__)
```

[ ]:
```
# load the model
model = load_model('model.h5')
```

[ ]:
```
# define a predict function as an endpoint
@app.route("/", methods=["POST"])
def predict():

    data = request.json
    data = np.asarray(data['input'])

    probs = model.predict(data)
    preds = probs.argmax(axis=-1)

    return class_names[preds[0]]
```

[ ]:
```
# start the flask app, allow remote connections
app.run(host='0.0.0.0')
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

1.1 Make a request to the model

- Run in a separate kernel

[ ]:
```
import json
data = json.dumps({'input': test_images[:1].tolist()})
```

[ ]:
```
import requests

headers = {"content-type": "application/json"}
json_response = requests.post('http://localhost:5000/', data=data,␣
 ↪headers=headers)
print('Prediction: ',json_response.text)
```

```
Prediction:  Ankle boot
```