Fashion MNIST

August 15, 2024

0.1 ANN on fashion MNIST dataset

Import Libraries

[14]:
```
pip install keras-tuner
```

```
Requirement already satisfied: keras-tuner in c:\users\saket
kumar\anaconda3\lib\site-packages (1.4.7)
Requirement already satisfied: keras in c:\users\saket kumar\anaconda3\lib\site-
packages (from keras-tuner) (3.3.3)
Requirement already satisfied: packaging in c:\users\saket
kumar\anaconda3\lib\site-packages (from keras-tuner) (23.2)
Requirement already satisfied: requests in c:\users\saket
kumar\anaconda3\lib\site-packages (from keras-tuner) (2.31.0)
Requirement already satisfied: kt-legacy in c:\users\saket
kumar\anaconda3\lib\site-packages (from keras-tuner) (1.0.5)
Requirement already satisfied: absl-py in c:\users\saket
kumar\anaconda3\lib\site-packages (from keras->keras-tuner) (2.1.0)
Requirement already satisfied: numpy in c:\users\saket kumar\anaconda3\lib\site-
packages (from keras->keras-tuner) (1.26.4)
Requirement already satisfied: rich in c:\users\saket kumar\anaconda3\lib\site-
packages (from keras->keras-tuner) (13.3.5)
Requirement already satisfied: namex in c:\users\saket kumar\anaconda3\lib\site-
packages (from keras->keras-tuner) (0.0.8)
Requirement already satisfied: h5py in c:\users\saket kumar\anaconda3\lib\site-
packages (from keras->keras-tuner) (3.11.0)
Requirement already satisfied: optree in c:\users\saket
kumar\anaconda3\lib\site-packages (from keras->keras-tuner) (0.11.0)
Requirement already satisfied: ml-dtypes in c:\users\saket
kumar\anaconda3\lib\site-packages (from keras->keras-tuner) (0.3.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\saket
kumar\anaconda3\lib\site-packages (from requests->keras-tuner) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\saket
kumar\anaconda3\lib\site-packages (from requests->keras-tuner) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\saket
kumar\anaconda3\lib\site-packages (from requests->keras-tuner) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\saket
kumar\anaconda3\lib\site-packages (from requests->keras-tuner) (2024.2.2)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\saket
kumar\anaconda3\lib\site-packages (from optree->keras->keras-tuner) (4.9.0)
```

```
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in c:\users\saket
kumar\anaconda3\lib\site-packages (from rich->keras->keras-tuner) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\saket
kumar\anaconda3\lib\site-packages (from rich->keras->keras-tuner) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\saket
kumar\anaconda3\lib\site-packages (from markdown-it-
py<3.0.0,>=2.2.0->rich->keras->keras-tuner) (0.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[15]:
```
pip show tensorboard
```

```
Name: tensorboardNote: you may need to restart the kernel to use updated
packages.

Version: 2.16.2
Summary: TensorBoard lets you watch Tensors Flow
Home-page: https://github.com/tensorflow/tensorboard
Author: Google Inc.
Author-email: packages@tensorflow.org
License: Apache 2.0
Location: C:\Users\Saket Kumar\anaconda3\Lib\site-packages
Requires: absl-py, grpcio, markdown, numpy, protobuf, setuptools, six,
tensorboard-data-server, werkzeug
Required-by: tensorflow-intel
```

[16]:
```
pip install tensorboard
```

```
Requirement already satisfied: tensorboard in c:\users\saket
kumar\anaconda3\lib\site-packages (2.16.2)
Requirement already satisfied: absl-py>=0.4 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (2.1.0)
Requirement already satisfied: grpcio>=1.48.2 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (1.62.1)
Requirement already satisfied: markdown>=2.6.8 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (3.4.1)
Requirement already satisfied: numpy>=1.12.0 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (1.26.4)
Requirement already satisfied: protobuf!=4.24.0,>=3.19.6 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (4.25.3)
Requirement already satisfied: setuptools>=41.0.0 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (68.2.2)
Requirement already satisfied: six>1.9 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (1.16.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
c:\users\saket kumar\anaconda3\lib\site-packages (from tensorboard) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\saket
kumar\anaconda3\lib\site-packages (from tensorboard) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\saket
kumar\anaconda3\lib\site-packages (from werkzeug>=1.0.1->tensorboard) (2.1.3)
```

```
Note: you may need to restart the kernel to use updated packages.
```

[17]:
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.callbacks import TensorBoard, CSVLogger, EarlyStopping,␣
  ↪ModelCheckpoint
from tensorflow.keras.initializers import HeNormal
from tensorflow.keras.regularizers import l2
from tensorflow.keras.models import load_model
import datetime
import keras_tuner as kt
```

Load and Prepare Data

[18]:
```
# Load Fashion MNIST dataset
(train_images, train_labels), (test_images, test_labels) = datasets.
  ↪fashion_mnist.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0

# Display some sample images
def plot_images(images, labels, class_names):
    plt.figure(figsize=(10,10))
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.imshow(images[i], cmap=plt.cm.binary)
        plt.xlabel(class_names[labels[i]])
    plt.show()

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

plot_images(train_images, train_labels, class_names)
```

Set Up Hyperparameter Tuning with Keras Tuner

```
[19]: def build_model(hp):
          model = models.Sequential()
          model.add(layers.Input(shape=(28, 28)))
          model.add(layers.Flatten())

          # Number of hidden layers
          num_hidden_layers = hp.Int('num_hidden_layers', min_value=1, max_value=3)

          for i in range(num_hidden_layers):
              units = hp.Int(f'units_{i}', min_value=64, max_value=128, step=32)
              model.add(layers.Dense(units=units,
                                     activation='relu',
                                     kernel_initializer=hp.
     ↪Choice('kernel_initializer', values=['glorot_uniform', 'he_normal']),
                                     kernel_regularizer=l2(hp.Float('l2',␣
     ↪min_value=1e-4, max_value=1e-2, sampling='LOG'))))
              model.add(layers.BatchNormalization())
              model.add(layers.Dropout(hp.Float('dropout_rate', min_value=0.1,␣
     ↪max_value=0.5, step=0.1)))

          model.add(layers.Dense(10, activation='softmax'))

          model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=hp.
     ↪Float('learning_rate', min_value=1e-4, max_value=1e-2, sampling='LOG')),
                        loss='sparse_categorical_crossentropy',
                        metrics=['accuracy'])
          return model
```

Set Up Callbacks

[20]:
```
# Set up TensorBoard
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)

# Set up CSVLogger
csv_logger = CSVLogger('training_log.csv', append=True)

# Set up EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=3)

# Set up ModelCheckpoint
model_checkpoint = ModelCheckpoint('best_model.keras', save_best_only=True)
```

Set Up Hyperparameter Tuning

[21]:
```
# Define the Keras Tuner
tuner = kt.RandomSearch(build_model,
                        objective='val_accuracy',
                        max_trials=5,
                        executions_per_trial=3,
                        directory='my_dir',
                        project_name='fashion_mnist')

# Perform hyperparameter tuning
tuner.search(train_images, train_labels, epochs=10,␣
↪validation_data=(test_images, test_labels))

# Get the best hyperparameters and build the model
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
```

```
model = tuner.hypermodel.build(best_hps)
```

```
Reloading Tuner from my_dir\fashion_mnist\tuner0.json
```

Train the Model

[22]:
```
# Train the model
history = model.fit(
    train_images, train_labels,
    epochs=10,
    validation_data=(test_images, test_labels),
    callbacks=[tensorboard_callback, csv_logger, early_stopping,␣
  ↪model_checkpoint]
)
```

```
Epoch 1/10
1875/1875                     11s 4ms/step -
accuracy: 0.7564 - loss: 0.7402 - val_accuracy: 0.8220 - val_loss: 0.5102
Epoch 2/10
1875/1875                     8s 4ms/step -
accuracy: 0.8470 - loss: 0.4494 - val_accuracy: 0.8405 - val_loss: 0.4565
Epoch 3/10
1875/1875                     7s 4ms/step -
accuracy: 0.8601 - loss: 0.4158 - val_accuracy: 0.8231 - val_loss: 0.5051
Epoch 4/10
1875/1875                     8s 4ms/step -
accuracy: 0.8652 - loss: 0.3972 - val_accuracy: 0.8619 - val_loss: 0.4231
Epoch 5/10
1875/1875                     7s 4ms/step -
accuracy: 0.8690 - loss: 0.3834 - val_accuracy: 0.8609 - val_loss: 0.4062
Epoch 6/10
1875/1875                     7s 4ms/step -
accuracy: 0.8735 - loss: 0.3696 - val_accuracy: 0.8625 - val_loss: 0.4034
Epoch 7/10
1875/1875                     7s 4ms/step -
accuracy: 0.8737 - loss: 0.3738 - val_accuracy: 0.8624 - val_loss: 0.4132
Epoch 8/10
1875/1875                     7s 4ms/step -
accuracy: 0.8738 - loss: 0.3638 - val_accuracy: 0.8581 - val_loss: 0.4134
Epoch 9/10
1875/1875                     7s 4ms/step -
accuracy: 0.8805 - loss: 0.3547 - val_accuracy: 0.8678 - val_loss: 0.3963
Epoch 10/10
1875/1875                     8s 4ms/step -
accuracy: 0.8778 - loss: 0.3523 - val_accuracy: 0.8581 - val_loss: 0.4324
```

Evaluate the Model

[23]:
```python
# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f'\nTest accuracy: {test_acc}')
```

```
313/313 - 1s - 2ms/step - accuracy: 0.8581 - loss: 0.4324

Test accuracy: 0.8580999970436096
```

Save and Load the Model

[24]:
```python
# Save the entire model
model.save('fashion_mnist_model.keras')

# Load the model
loaded_model = load_model('fashion_mnist_model.keras')
```

Visualize Training Process with TensorBoard

[25]:
```python
import subprocess

# Define the log directory
log_dir = 'logs/fit'
port = 6006  # Define the port number

# Start TensorBoard
def start_tensorboard(log_dir, port):
    try:
        # Start TensorBoard
        tensorboard_command = f'tensorboard --logdir={log_dir} --port={port}'
        process = subprocess.Popen(tensorboard_command, shell=True)
        print(f"Started TensorBoard on port {port}.")
    except Exception as e:
        print(f"Error while starting TensorBoard: {e}")

# Start TensorBoard
start_tensorboard(log_dir, port)
```

```
Started TensorBoard on port 6006.
```

Hyperparameter Tuning

[26]:
```python
# Print the best hyperparameters
print(f'Best learning rate: {best_hps.get("learning_rate")}')
print(f'Best number of hidden layers: {best_hps.get("num_hidden_layers")}')
for i in range(best_hps.get("num_hidden_layers")):
    print(f'Best number of units in hidden layer {i+1}: {best_hps.
  ↪get(f"units_{i}")}')
print(f'Best dropout rate: {best_hps.get("dropout_rate")}')
print(f'Best kernel initializer: {best_hps.get("kernel_initializer")}')
print(f'Best L2 regularization: {best_hps.get("l2")}')
```

```
Best learning rate: 0.00047408017755123694
Best number of hidden layers: 1
Best number of units in hidden layer 1: 64
Best dropout rate: 0.1
Best kernel initializer: he_normal
Best L2 regularization: 0.00013975903982386042
```

```
[ ]:
```