ANN_For_image_Classification

June 18, 2025

[5]:
```
# Install Tensor flow

!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-
packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (24.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3
in /usr/local/lib/python3.11/dist-packages (from tensorflow) (5.29.5)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (4.14.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
```

```
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.73.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (3.14.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.45.1)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
(3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2025.4.26)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.11/dist-packages (from
tensorboard<2.19,>=2.18->tensorflow) (3.8)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.11/dist-packages (from
tensorboard<2.19,>=2.18->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from
tensorboard<2.19,>=2.18->tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.11/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow)
(3.0.0)
```

```
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow)
(2.19.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-
packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
```

[4]:
```python
import tensorflow as tf
```

[11]:
```python
print(tf.__version__)
```

```
2.18.0
```

[3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```python

```

1 Data Preprocessing

[2]:
```python
from tensorflow.keras.datasets import fashion_mnist
```

[8]:
```python
# Loading the data set

(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

[9]:
```python
x_train.shape
```

[9]:
```
(60000, 28, 28)
```

[10]:
```python
x_test.shape
```

[10]:
```
(10000, 28, 28)
```

[11]:
```python
y_train.shape
```

[11]:
```
(60000,)
```

[12]:
```python
y_test.shape
```

[12]:
```
(10000,)
```

[13]:
```python
x_train
```

[13]:
```
array([[[0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]],

       [[0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]],

       [[0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]],

       …,

       [[0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]],

       [[0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]],

       [[0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]]], dtype=uint8)
```

[17]:
```
np.max(x_train), np.min(x_test) , np.mean(x_train)
```

[17]:
```
(np.uint8(255), np.uint8(0), np.float64(72.94035223214286))
```

[18]:
```
y_train
```

[18]:
```
array([9, 0, 0, …, 3, 0, 5], dtype=uint8)
```

[19]:
```
np.max(y_train), np.min(y_train) , np.mean(y_train)
```

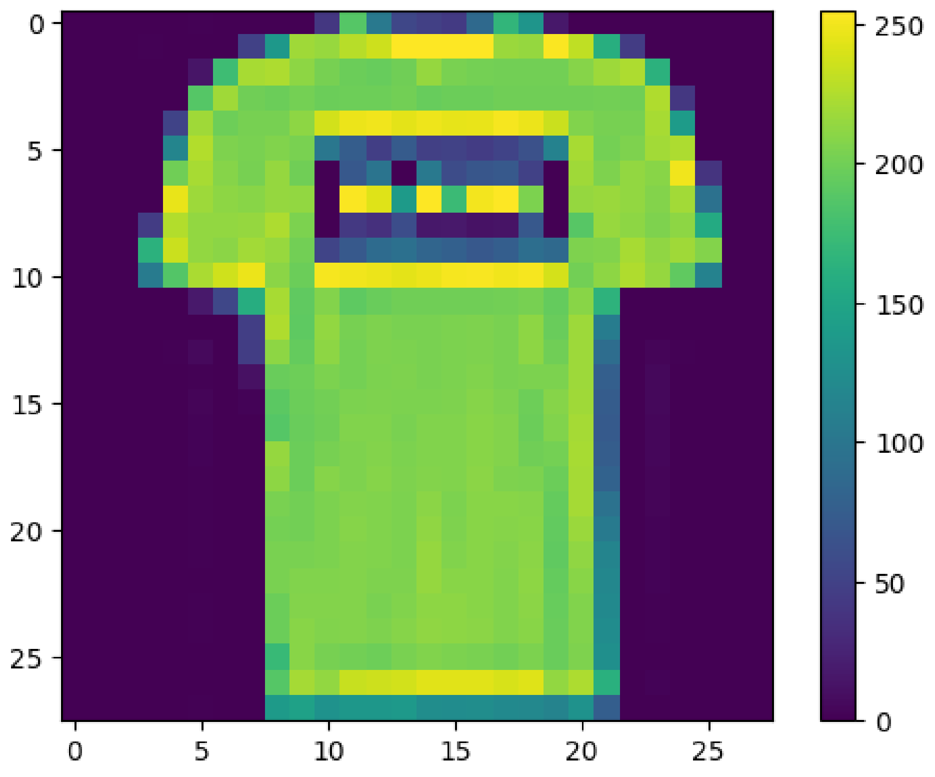
[19]:
```
(np.uint8(9), np.uint8(0), np.float64(4.5))
```

[20]:
```
class_names = ['0 T-shirt/top', '1 Trouser', '2 Pullover', '3 Dress', '4 Coat',
               '5 Sandal', '6 Shirt', '7 Sneaker', '8 Bag', '9 Ankle boot']
print(class_names)
```

```
['0 T-shirt/top', '1 Trouser', '2 Pullover', '3 Dress', '4 Coat', '5 Sandal', '6
Shirt', '7 Sneaker', '8 Bag', '9 Ankle boot']
```

[23]:
```
# Data Exploration
plt.figure()
plt.imshow(x_train[1])
plt.colorbar()
```

[23]:
```
<matplotlib.colorbar.Colorbar at 0x7e8e90b14c90>
```



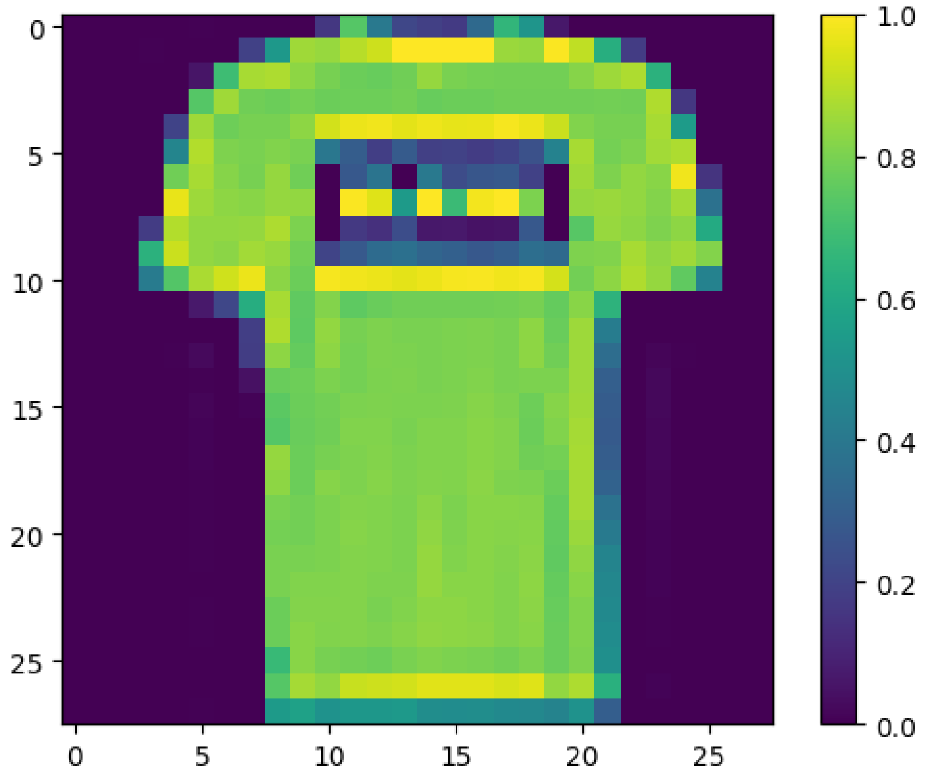
[22]:
```
y_train[1]
```

[22]:
```
np.uint8(0)
```

[24]:
```
# Normalising the dataset after normalization neural networks learn faster
x_train = x_train/255.0
x_test = x_test/255.0
```

[25]:
```
plt.figure()
plt.imshow(x_train[1])
plt.colorbar()
```

[25]:
```
<matplotlib.colorbar.Colorbar at 0x7e8e90b2fd90>
```



[26]:
```
# Flattening the dataset
x_train.shape, x_test.shape
```

[26]:
```
((60000, 28, 28), (10000, 28, 28))
```

[27]:
```
x_train = x_train.reshape(-1, 28*28)
x_test = x_test.reshape(-1, 28*28)
```

```
[36]: x_test.shape, x_train.shape
```

```
[36]: ((10000, 784), (60000, 784))
```

```
[30]: # define the model
      model = tf.keras.models.Sequential()
      # sequence of layers
```

```
[33]: # add fully connected hidden layer
      # 1) no.of neurons = 128 i.e units
      # 2) activation function = ReLu
      # 3) input shape = 784 (flattern dataset) i.e input shape vector
      #model.add(tf.keras.layers.Dense(units=128, activation='relu',␣
       ↪input_shape=(784,)))
      model = tf.keras.Sequential([
          tf.keras.Input(shape=(784,)),
          tf.keras.layers.Dense(units=128, activation='relu')
      ])
```

```
[34]: # Adding the second layer with dropout
      model.add(tf.keras.layers.Dropout(0.3))
      # the regularization technique can prevents overfitting
```

```
[40]: # Adding the output layer
      # units = 10 no. f nuerons i.e there are 10 classes in our outputs
      # activation function = softmax for multiple output softmax is used
      model.add(tf.keras.layers.Dense(units = 10, activation = 'softmax'))
```

```
[41]: # Compiling the model
      # 1) Optimizer = adam, to minimize the loss function
      # 2) loss function = sparse_categorical_crossentropy acts as guide to optimizer
      # 3) matrices = sparse_categorical_accuracy
```

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

```
[42]: model.summary()
```

```
Model: "sequential_1"


 Layer (type)                         Output Shape                   Param #

 dense_1 (Dense)                      (None, 128)                    100,480
 dropout (Dropout)                  (None, 128)                         0

 dense_2 (Dense)                    (None, 10)                      1,290

 dense_3 (Dense)                    (None, 10)                        110


 Total params: 101,880 (397.97 KB)

 Trainable params: 101,880 (397.97 KB)

 Non-trainable params: 0 (0.00 B)
```

[43]:
```
model.fit(x_train, y_train, epochs=10)
```

```
Epoch 1/10
1875/1875            8s 3ms/step -
loss: 1.7521 - sparse_categorical_accuracy: 0.4218
Epoch 2/10
1875/1875            10s 3ms/step -
loss: 1.1655 - sparse_categorical_accuracy: 0.5465
Epoch 3/10
1875/1875            11s 4ms/step -
loss: 0.9920 - sparse_categorical_accuracy: 0.6057
Epoch 4/10
1875/1875            8s 4ms/step -
loss: 0.8860 - sparse_categorical_accuracy: 0.6449
Epoch 5/10
1875/1875            7s 3ms/step -
loss: 0.8272 - sparse_categorical_accuracy: 0.6531
Epoch 6/10
1875/1875            11s 4ms/step -
loss: 0.7602 - sparse_categorical_accuracy: 0.7103
Epoch 7/10
1875/1875            11s 4ms/step -
loss: 0.6191 - sparse_categorical_accuracy: 0.8032
Epoch 8/10
1875/1875            7s 4ms/step -
loss: 0.5370 - sparse_categorical_accuracy: 0.8416
Epoch 9/10
1875/1875            10s 3ms/step -
loss: 0.4901 - sparse_categorical_accuracy: 0.8492
Epoch 10/10
1875/1875            8s 4ms/step -
loss: 0.4672 - sparse_categorical_accuracy: 0.8559
```

[43]:
```
<keras.src.callbacks.history.History at 0x7e8e6e5cd3d0>
```

[45]:
```
# Evaluate the model on the test dataset
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print('Test Accuracy: {}'.format(test_accuracy))
```

```
313/313              1s 2ms/step -
loss: 0.4652 - sparse_categorical_accuracy: 0.8613
Test Accuracy: 0.8583999872207642
```

[47]:
```
# Model prediction
import numpy as np

# Model prediction
y_prob = model.predict(x_test) # This will return probabilities for each class
y_pred = np.argmax(y_prob, axis=-1) # This converts probabilities to class␣
 ↪labels

print(y_pred)
```

```
313/313              1s 2ms/step
[9 2 1 … 8 1 5]
```

[48]:
```
y_pred[0]
```

[48]:
```
np.int64(9)
```

[49]:
```
y_test[0]
```

[49]:
```
np.uint8(9)
```

[50]:
```
y_pred[110] , y_test[110]
```

[50]:
```
(np.int64(2), np.uint8(2))
```

[51]:
```
print(class_names)
```

```
['0 T-shirt/top', '1 Trouser', '2 Pullover', '3 Dress', '4 Coat', '5 Sandal', '6
Shirt', '7 Sneaker', '8 Bag', '9 Ankle boot']
```

[52]:
```
# Confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test, y_pred)

print(cm)
```

```
[[872   2   6  26   3   0  82   0   9   0]
 [  1 963   0  22   6   0   7   0   1   0]
 [  9   0 738  11 104   1 134   0   3   0]
 [ 26   5   5 894  29   0  36   0   5   0]
 [  1   0 167  52 713   0  66   0   1   0]
 [  0   0   0   0   0 952   0  25   2  21]
 [210   1  83  29  66   0 600   0  11   0]
 [  0   0   0   0   0  25   0 921   0  54]
 [  1   0   7   6   2   4  13   7 960   0]
 [  1   0   0   0   0   4   0  24   0 971]]
```

[53]:
```
acc_cm = accuracy_score(y_test, y_pred)
print(acc_cm)
```

```
0.8584
```