ddef8fxgt

August 26, 2024

[ ]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist, cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import SGD
import random
```

[ ]:

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

[ ]:

```
#Normalize the images to the range [0,1
x_train,x_test = x_train / 255.0, x_test / 255.0
```

[ ]:

```
y_train = tf.keras.utils.to_categorical(y_train,10)
y_test = tf.keras.utils.to_categorical(y_test,10
```

[ ]:

```
model = Sequential([
    Flatten(input_shape=(28,28)),
    Dense(128,activation='relu'),
    Dense(64,activation='relu'),
    Dense(10,activation='softmax')
])
```

[ ]:

```
#Compile the model
model.compile(optimizer=SGD(),
             loss='categorical_crossentropy',
             metrics=['accuracy'])
```

[ ]:

```
#TRain the model
history = model.fit(x_train,y_train,epochs=20,␣
 ↪batch_size=32,validation_data=(x_test,y_test))
```

```
Epoch 1/20
```

```
2024-08-06 09:49:40.637986: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 188160000 exceeds 10% of free system memory.
```

```
1875/1875 [==============================] - 2s 912us/step - loss: 0.6835 -
accuracy: 0.8156 - val_loss: 0.3269 - val_accuracy: 0.9061
Epoch 2/20
1875/1875 [==============================] - 2s 858us/step - loss: 0.2977 -
accuracy: 0.9141 - val_loss: 0.2565 - val_accuracy: 0.9271
Epoch 3/20
1875/1875 [==============================] - 2s 853us/step - loss: 0.2432 -
accuracy: 0.9299 - val_loss: 0.2162 - val_accuracy: 0.9366
Epoch 4/20
1875/1875 [==============================] - 2s 881us/step - loss: 0.2092 -
accuracy: 0.9392 - val_loss: 0.1887 - val_accuracy: 0.9453
Epoch 5/20
1875/1875 [==============================] - 2s 854us/step - loss: 0.1840 -
accuracy: 0.9460 - val_loss: 0.1696 - val_accuracy: 0.9516
Epoch 6/20
1875/1875 [==============================] - 2s 857us/step - loss: 0.1642 -
accuracy: 0.9517 - val_loss: 0.1574 - val_accuracy: 0.9539
Epoch 7/20
1875/1875 [==============================] - 2s 838us/step - loss: 0.1477 -
accuracy: 0.9575 - val_loss: 0.1427 - val_accuracy: 0.9572
Epoch 8/20
1875/1875 [==============================] - 2s 909us/step - loss: 0.1344 -
accuracy: 0.9607 - val_loss: 0.1321 - val_accuracy: 0.9611
Epoch 9/20
1875/1875 [==============================] - 2s 865us/step - loss: 0.1231 -
accuracy: 0.9646 - val_loss: 0.1259 - val_accuracy: 0.9624
Epoch 10/20
1875/1875 [==============================] - 2s 857us/step - loss: 0.1137 -
accuracy: 0.9675 - val_loss: 0.1172 - val_accuracy: 0.9647
Epoch 11/20
1875/1875 [==============================] - 2s 847us/step - loss: 0.1057 -
accuracy: 0.9699 - val_loss: 0.1145 - val_accuracy: 0.9648
Epoch 12/20
1875/1875 [==============================] - 2s 851us/step - loss: 0.0983 -
accuracy: 0.9717 - val_loss: 0.1095 - val_accuracy: 0.9675
Epoch 13/20
1875/1875 [==============================] - 2s 850us/step - loss: 0.0921 -
accuracy: 0.9737 - val_loss: 0.1064 - val_accuracy: 0.9678
Epoch 14/20
1875/1875 [==============================] - 2s 857us/step - loss: 0.0862 -
accuracy: 0.9754 - val_loss: 0.1026 - val_accuracy: 0.9688
Epoch 15/20
1875/1875 [==============================] - 2s 858us/step - loss: 0.0814 -
accuracy: 0.9769 - val_loss: 0.0980 - val_accuracy: 0.9698
Epoch 16/20
1875/1875 [==============================] - 2s 849us/step - loss: 0.0765 -
accuracy: 0.9776 - val_loss: 0.0980 - val_accuracy: 0.9697
Epoch 17/20
```

```
1875/1875 [==============================] - 2s 851us/step - loss: 0.0723 -
accuracy: 0.9793 - val_loss: 0.0902 - val_accuracy: 0.9722
Epoch 18/20
1875/1875 [==============================] - 2s 859us/step - loss: 0.0683 -
accuracy: 0.9804 - val_loss: 0.0892 - val_accuracy: 0.9728
Epoch 19/20
1875/1875 [==============================] - 2s 874us/step - loss: 0.0646 -
accuracy: 0.9812 - val_loss: 0.0898 - val_accuracy: 0.9716
Epoch 20/20
1875/1875 [==============================] - 2s 849us/step - loss: 0.0614 -
accuracy: 0.9827 - val_loss: 0.0864 - val_accuracy: 0.9729
```

[ ]:

```python
#Evaluating
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'test_loss: {test_loss}')
print(f'Test Accuracy: {test_acc}')
```

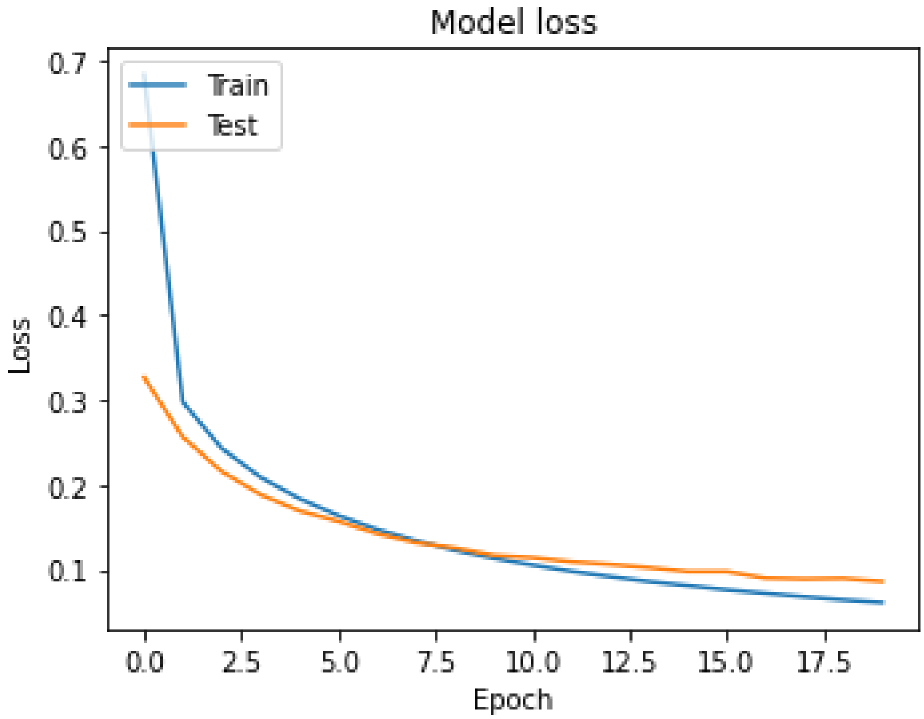
```
313/313 [==============================] - 0s 604us/step - loss: 0.0864 -
accuracy: 0.9729
test_loss: 0.08636705577373505
Test Accuracy: 0.9728999733924866
```

[ ]:

```python
#Plotting training loss and accuracy
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend (['Train','Test'],loc='upper left')
```

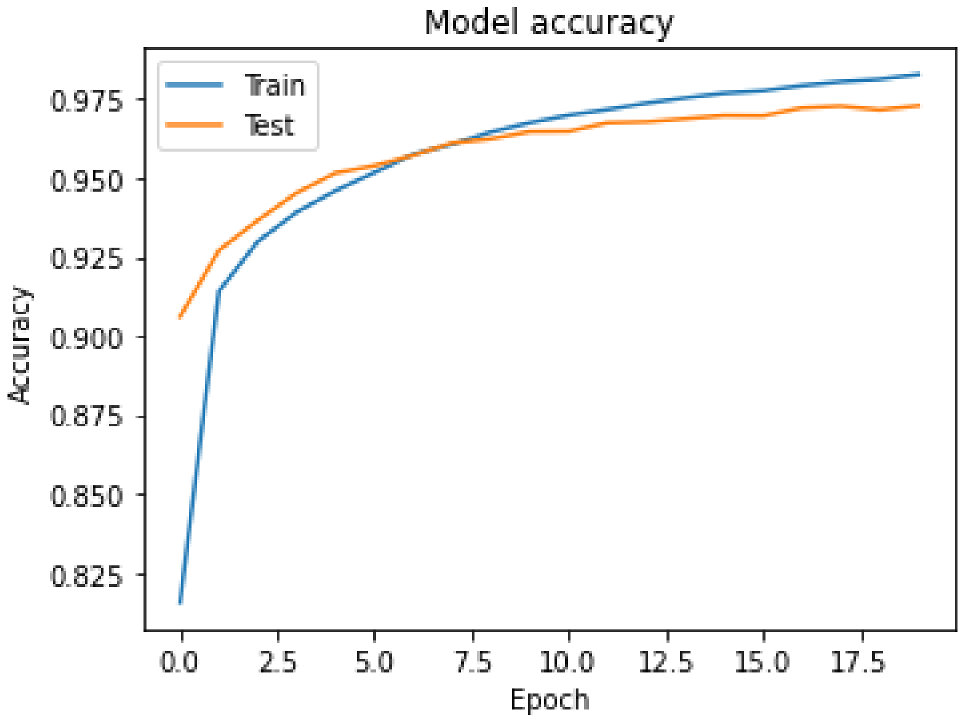
[ ]:

```
<matplotlib.legend.Legend at 0x7fdffccb5ed0>
```



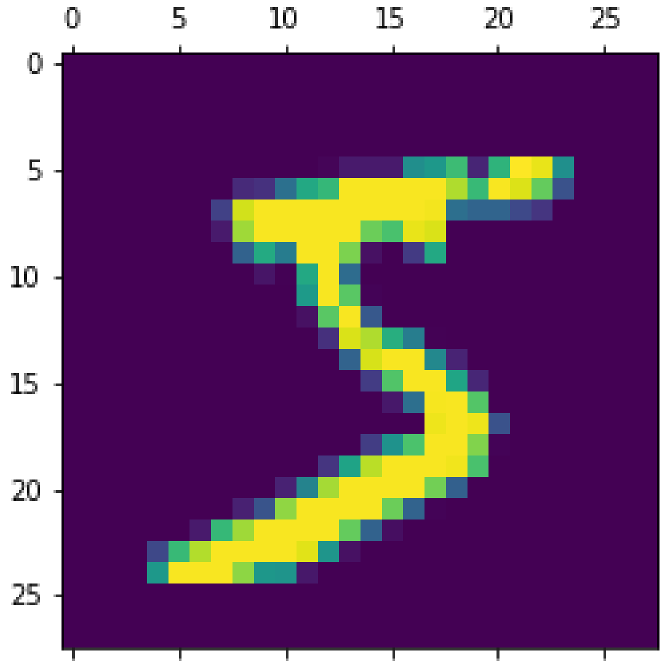
[ ]:

```
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend (['Train','Test'],loc='upper left')
plt.show()
```



```
[ ]: plt.matshow(x_train[0])
```

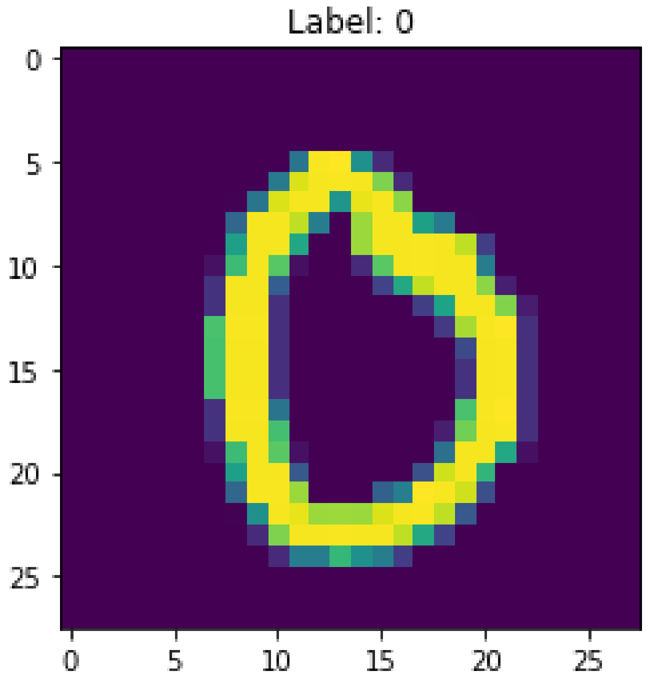
```
[ ]: <matplotlib.image.AxesImage at 0x7fdfd47e0ac0>
```



[ ]:

```
n = random.randint(0, len(x_test) - 1)

# Display the image
plt.imshow(x_test[n],)
plt.title(f"Label: {np.argmax(y_test[n])}")
plt.show()
```



[ ]:

[ ]: