DSC550_WEEK11

April 5, 2025

[41]:
```
#Import libraries
```

[42]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
import seaborn as sns
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import confusion_matrix
```

[43]:
```
#Load the MNIST data set
```

[44]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print("Training Data Shape:", x_train.shape)
print("Training Labels Shape:", y_train.shape)
print("Test Data Shape:", x_test.shape)
print("Test Labels Shape:", y_test.shape)
```

```
Training Data Shape: (60000, 28, 28)
Training Labels Shape: (60000,)
Test Data Shape: (10000, 28, 28)
Test Labels Shape: (10000,)
```

[45]:
```
#Display the first five images in the training data set
```

[46]:
```
(x_train, y_train), (_, _) = mnist.load_data()

for i in range(5):
    plt.subplot(1, 5, i + 1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')

plt.show()
```

[47]:
```
#Build and train a Keras CNN classifier on the MNIST training set
```

[48]:
```
x_train = x_train.reshape((60000, 28, 28, 1)).astype('float32') / 255
x_test = x_test.reshape((10000, 28, 28, 1)).astype('float32') / 255
y_train = tf.keras.utils.to_categorical(y_train)
y_test = tf.keras.utils.to_categorical(y_test)


model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))


model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=5, batch_size=64,␣
 ↪validation_split=0.2)

test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc}")
```

```
Epoch 1/5
750/750 [==============================] - 8s 11ms/step - loss: 0.2090 -
accuracy: 0.9374 - val_loss: 0.0660 - val_accuracy: 0.9816
Epoch 2/5
750/750 [==============================] - 8s 11ms/step - loss: 0.0607 -
accuracy: 0.9815 - val_loss: 0.0500 - val_accuracy: 0.9852
Epoch 3/5
750/750 [==============================] - 8s 11ms/step - loss: 0.0429 -
accuracy: 0.9860 - val_loss: 0.0472 - val_accuracy: 0.9863
```

```
Epoch 4/5
750/750 [==============================] - 8s 11ms/step - loss: 0.0317 -
accuracy: 0.9902 - val_loss: 0.0422 - val_accuracy: 0.9873
Epoch 5/5
750/750 [==============================] - 8s 11ms/step - loss: 0.0268 -
accuracy: 0.9912 - val_loss: 0.0454 - val_accuracy: 0.9868
313/313 [==============================] - 1s 2ms/step - loss: 0.0322 -
accuracy: 0.9906
Test Accuracy: 0.9905999898910522
```

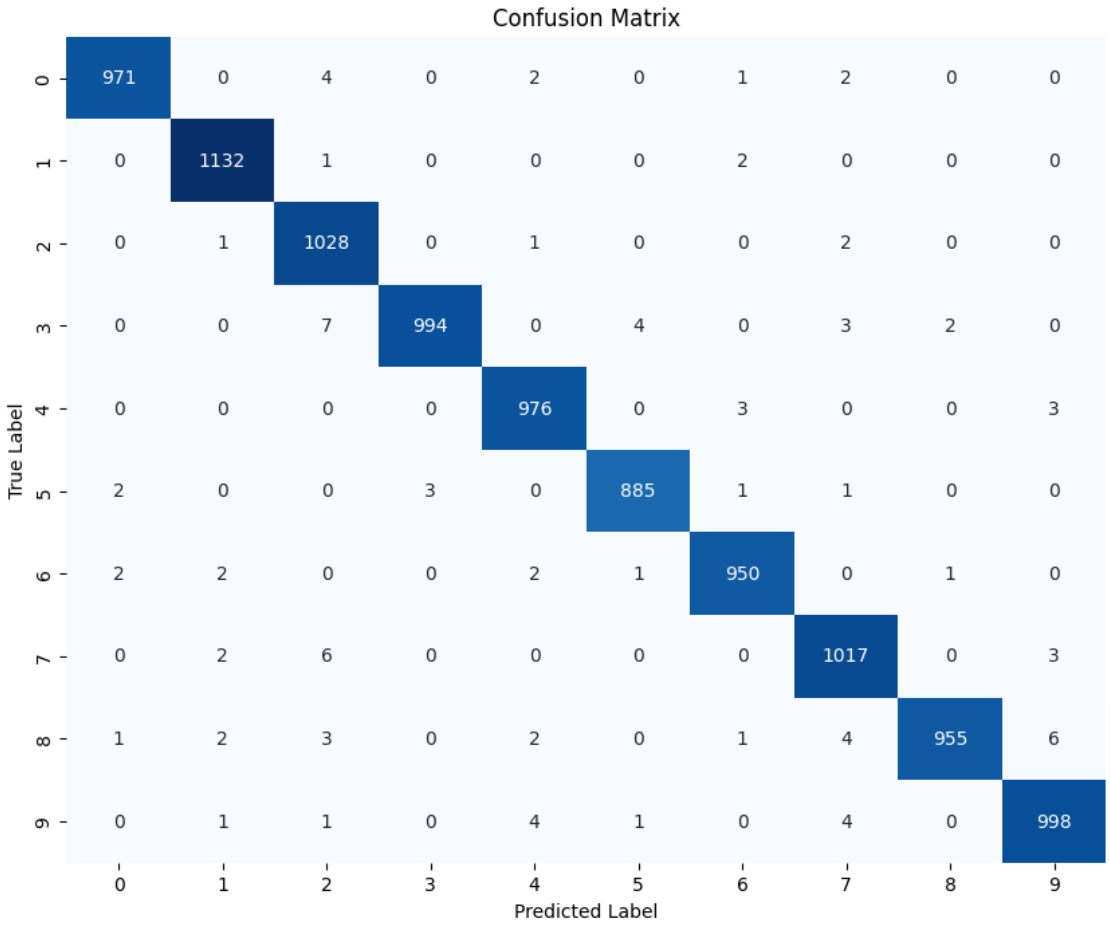
[49]:
```
#Confusion matrix on test
```

[50]:
```
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)

conf_matrix = confusion_matrix(y_test_classes, y_pred_classes)

plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=list(range(10)), yticklabels=list(range(10)))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
313/313 [==============================] - 1s 2ms/step
```


Confusion Matrix
True Label
Predicted Label
0 971 0 4 0 2 0 1 2 0 0
1 0 1132 1 0 0 0 2 0 0 0
2 0 1 1028 0 1 0 0 2 0 0
3 0 0 7 994 0 4 0 3 2 0
4 0 0 0 0 976 0 3 0 0 3
5 2 0 0 3 0 885 1 1 0 0
6 2 2 0 0 2 1 950 0 1 0
7 0 2 6 0 0 0 0 1017 0 3
8 1 2 3 0 2 0 1 4 955 6
9 0 1 1 0 4 1 0 4 0 998
0 1 2 3 4 5 6 7 8 9

[ ]: