Brian_Reppeto_DSC550_Week_11

May 26, 2024

0.0.1 DSC 550 Week :

Activity 11.2

Author: Brian Reppeto 5/22/2024

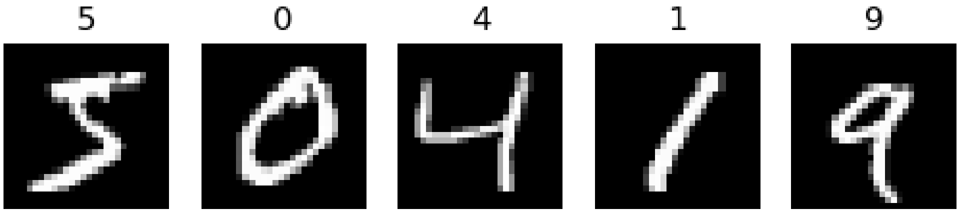
[12]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

[13]:
```
# load the MNIST dataset

(X_train, y_train),(X_test, y_test) = mnist.load_data()
```

[14]:
```
# display the first five images

for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(X_train[i], cmap='gray')
    plt.title(y_train[i])
    plt.axis('off')
plt.show()
```



[15]:
```
# process the data
```

```
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32') / 255
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32') / 255
```

[16]:

```
# one-hot encode the labels

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

[17]:

```
# build the CNN model

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

[18]:

```
# compile the model

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

[19]:

```
# train the model

model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10,␣
 ↪batch_size=200, verbose=2)
```

```
Epoch 1/10
300/300 - 6s - loss: 0.2475 - accuracy: 0.9294 - val_loss: 0.0602 -
val_accuracy: 0.9823 - 6s/epoch - 19ms/step
Epoch 2/10
300/300 - 6s - loss: 0.0623 - accuracy: 0.9812 - val_loss: 0.0450 -
val_accuracy: 0.9873 - 6s/epoch - 19ms/step
Epoch 3/10
300/300 - 6s - loss: 0.0441 - accuracy: 0.9866 - val_loss: 0.0341 -
val_accuracy: 0.9889 - 6s/epoch - 19ms/step
Epoch 4/10
300/300 - 6s - loss: 0.0351 - accuracy: 0.9896 - val_loss: 0.0386 -
val_accuracy: 0.9874 - 6s/epoch - 19ms/step
Epoch 5/10
300/300 - 6s - loss: 0.0282 - accuracy: 0.9909 - val_loss: 0.0296 -
val_accuracy: 0.9903 - 6s/epoch - 19ms/step
Epoch 6/10
300/300 - 6s - loss: 0.0218 - accuracy: 0.9934 - val_loss: 0.0266 -
val_accuracy: 0.9918 - 6s/epoch - 19ms/step
```

```
Epoch 7/10
300/300 - 6s - loss: 0.0188 - accuracy: 0.9942 - val_loss: 0.0287 -
val_accuracy: 0.9907 - 6s/epoch - 19ms/step
Epoch 8/10
300/300 - 6s - loss: 0.0148 - accuracy: 0.9953 - val_loss: 0.0336 -
val_accuracy: 0.9897 - 6s/epoch - 19ms/step
Epoch 9/10
300/300 - 6s - loss: 0.0123 - accuracy: 0.9959 - val_loss: 0.0266 -
val_accuracy: 0.9922 - 6s/epoch - 19ms/step
Epoch 10/10
300/300 - 6s - loss: 0.0104 - accuracy: 0.9966 - val_loss: 0.0329 -
val_accuracy: 0.9896 - 6s/epoch - 19ms/step
```

[19]:
```
<keras.callbacks.History at 0x313e6dc90>
```

[20]:
```python
# evaluate the model

test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Test accuracy: {test_accuracy:.4f}')
```

```
Test accuracy: 0.9896
```

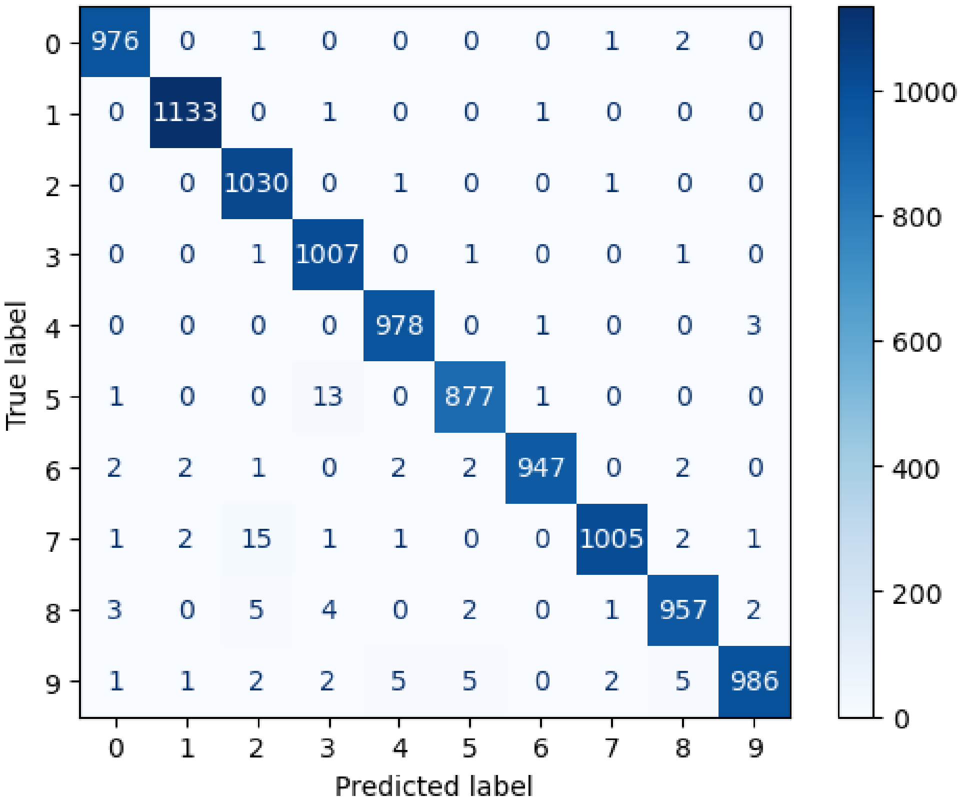
[21]:
```python
# predict the test set

y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
313/313 [==============================] - 1s 3ms/step
```

[22]:
```python
# confusion matrix

cm = confusion_matrix(y_true, y_pred_classes)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=range(10))
disp.plot(cmap=plt.cm.Blues)
plt.show()
```



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	976	0	1	0	0	0	0	1	2	0
1	0	1133	0	1	0	0	1	0	0	0
2	0	0	1030	0	1	0	0	1	0	0
3	0	0	1	1007	0	1	0	0	1	0
4	0	0	0	0	978	0	1	0	0	3
5	1	0	0	13	0	877	1	0	0	0
6	2	2	1	0	2	2	947	0	2	0
7	1	2	15	1	1	0	0	1005	2	1
8	3	0	5	4	0	2	0	1	957	2
9	1	1	2	2	5	5	0	2	5	986

The CNN image classifier trained on the MNIST dataset achieved a test accuracy of approximately 99%. The high accuracy indicates that the model is effective at recognizing handwritten digits. The confusion matrix further illustrates the model's performance, showing a high number of correct predictions across all digit classes. There were a few misclassifications, which demonstrates the model's robustness in distinguishing between different digits.

Overall, the model performs exceptionally well, making it a reliable tool for digit recognition tasks.

[ ]:

[ ]: