MNIST Digit Classification with CNN

June 23, 2025

[4]:
```
#MNIST Digit Classification with CNN (TensorFlow)

import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np
```

[5]:
```
#Load MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434              0s
0us/step
```

[6]:
```
# Preprocess the data
# Reshape to (28, 28, 1) and normalize pixel values to [0, 1]
x_train = x_train.reshape(-1, 28, 28, 1).astype("float32") / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype("float32") / 255.0
```

[8]:
```
#Build CNN model
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')  # 10 digits (0-9)
])
```

[9]:
```
#Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[10]:
```
#Train the model
model.fit(x_train, y_train, epochs=5, validation_split=0.1)
```

```
Epoch 1/5
1688/1688             22s 12ms/step -
accuracy: 0.8947 - loss: 0.3469 - val_accuracy: 0.9855 - val_loss: 0.0477
Epoch 2/5
1688/1688             21s 12ms/step -
accuracy: 0.9836 - loss: 0.0518 - val_accuracy: 0.9892 - val_loss: 0.0379
Epoch 3/5
1688/1688             24s 14ms/step -
accuracy: 0.9903 - loss: 0.0316 - val_accuracy: 0.9898 - val_loss: 0.0362
Epoch 4/5
1688/1688             20s 12ms/step -
accuracy: 0.9923 - loss: 0.0247 - val_accuracy: 0.9872 - val_loss: 0.0433
Epoch 5/5
1688/1688             30s 18ms/step -
accuracy: 0.9945 - loss: 0.0171 - val_accuracy: 0.9927 - val_loss: 0.0327
```

[10]:
```
<keras.src.callbacks.history.History at 0x7018b3fe63b0>
```

[11]:
```
#Evaluate on test set
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Accuracy: {:.2f}%".format(test_acc * 100))
```

```
313/313             1s 4ms/step -
accuracy: 0.9895 - loss: 0.0357
Test Accuracy: 99.10%
```

[13]:
```
#Visualize predictions on 5 sample test images
predictions = model.predict(x_test[:5])
predicted_labels = np.argmax(predictions, axis=1)
plt.figure(figsize=(10, 2))
for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.title(f"Pred: {predicted_labels[i]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

```
1/1             0s 27ms/step
```