MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

**Volodymyr Shymkovych**

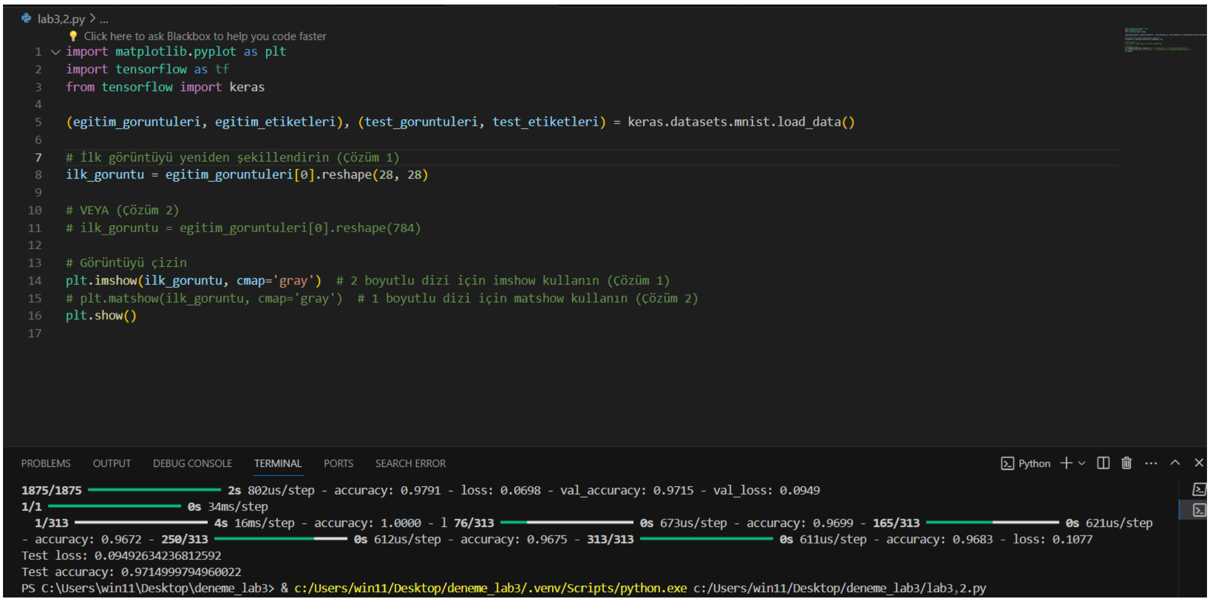
# Design and implementation of software systems with neural networks

**LABORATORY WORK #3**

**Training and Evaluating Neural Networks for MNIST Digit Classification Using Keras**

kulubecioglu mehmet
IM-14 FIOT

Kyiv
IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE
2024



lab3,2.py > ...

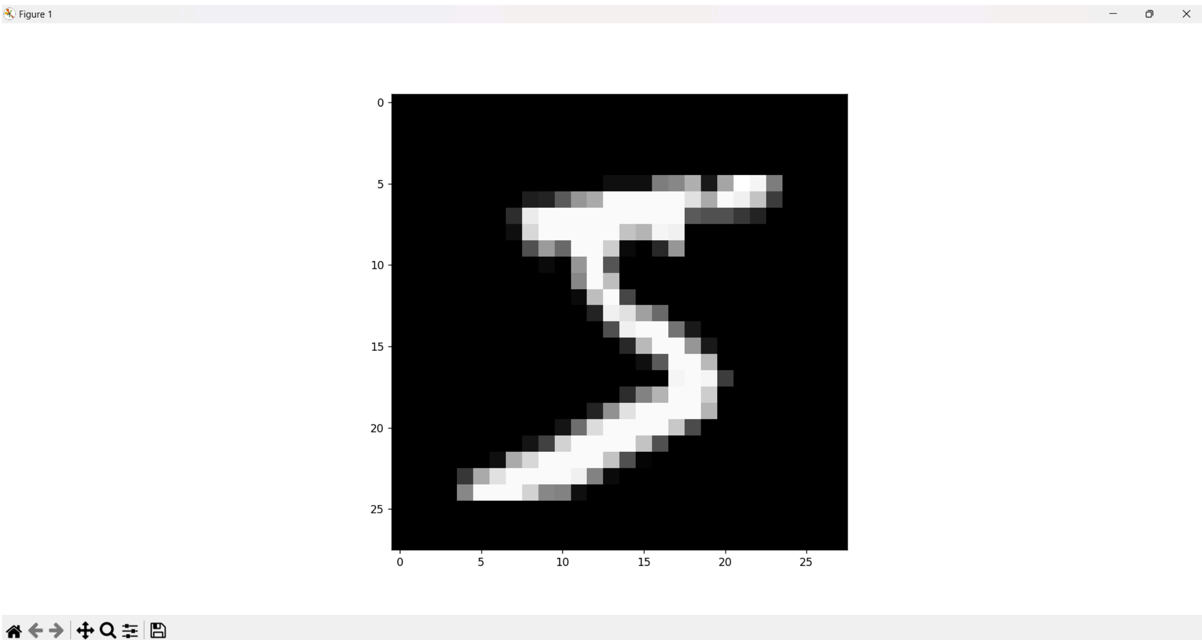
Click here to ask Blackbox to help you code faster

```python
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras

(egitim_goruntuleri, egitim_etiketleri), (test_goruntuleri, test_etiketleri) = keras.datasets.mnist.load_data()

# İlk görüntüyü yeniden şekillendirin (Çözüm 1)
ilk_goruntu = egitim_goruntuleri[0].reshape(28, 28)

# VEYA (Çözüm 2)
# ilk_goruntu = egitim_goruntuleri[0].reshape(784)

# Görüntüyü çizin
plt.imshow(ilk_goruntu, cmap='gray')  # 2 boyutlu dizi için imshow kullanın (Çözüm 1)
# plt.matshow(ilk_goruntu, cmap='gray')  # 1 boyutlu dizi için matshow kullanın (Çözüm 2)
plt.show()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR Python

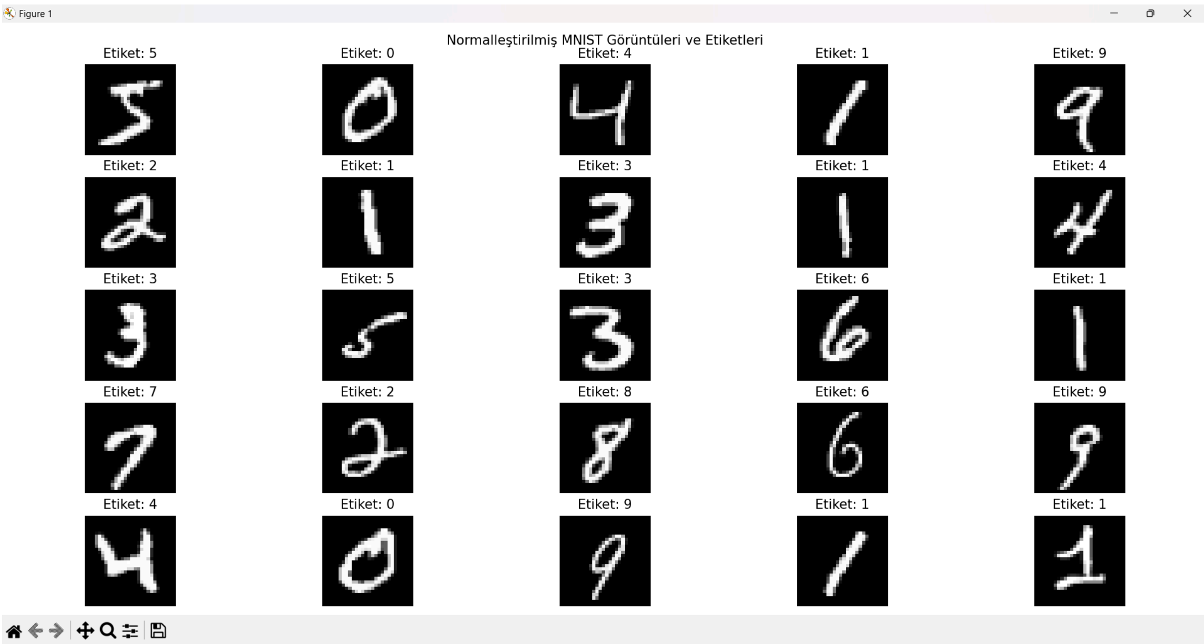
```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 802us/step - accuracy: 0.9791 - loss: 0.0698 - val_accuracy: 0.9715 - val_loss: 0.0949
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
  1/313 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - accuracy: 1.0000 - l 76/313 ━━━━━━━━━━━━━━━━━━━━ 0s 673us/step - accuracy: 0.9699 - 165/313 ━━━━━━━━━━━━━━━━━━━━ 0s 621us/step
- accuracy: 0.9672 - 250/313 ━━━━━━━━━━━━━━━━━━━━ 0s 612us/step - accuracy: 0.9675 - 313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 611us/step - accuracy: 0.9683 - loss: 0.1077
Test loss: 0.09492634236812592
Test accuracy: 0.9714999794960022
PS C:\Users\win11\Desktop\deneme_lab3> & c:/Users/win11/Desktop/deneme_lab3/.venv/Scripts/python.exe c:/Users/win11/Desktop/deneme_lab3/lab3,2.py
```

This code loads a sample image from the MNIST dataset, reshapes it, and visualizes it. Firstly, it imports the necessary libraries: **`matplotlib.pyplot`** and **`tensorflow.keras`**. Then, it loads the MNIST dataset using the **`mnist.load_data()`** function, separating training/test images and labels. The code offers two solutions: Solution 1 reshapes the image into a 28x28 array, while Solution 2 converts the same array into a 784-dimensional flattened array. Finally, the image is visualized using **`plt.imshow()`**.



```python
import matplotlib.pyplot as plt
import numpy as np
from tensorflow import keras
# MNIST verilerini yükleyin
(egitim_goruntuleri, egitim_etiketleri), (test_goruntuleri, test_etiketleri) = keras.datasets.mnist.load_data()

# Verileri normalleştirin
egitim_goruntuleri_normal = egitim_goruntuleri.astype(np.float32) / 255.0

# İlk 25 görüntüyü ve etiketlerini seçin
goruntuler = egitim_goruntuleri_normal[0:25]
etiketler = egitim_etiketleri[0:25]

# Görüntüleri Görüntüle
fig, axes = plt.subplots(5, 5, figsize=(10, 10))

for i, (goruntu, etiket) in enumerate(zip(goruntuler, etiketler)):
    # Görüntüyü düzleştirin
    goruntu_duz = goruntu.flatten()

    # Alt grafiğe geçin
    axes[int(i / 5), i % 5].imshow(goruntu, cmap='gray')

    # Etiketleri görüntüye ekleyin
    axes[int(i / 5), i % 5].set_title(f"Etiket: {etiket}")

    # Eksenleri gizleyin
    axes[int(i / 5), i % 5].axis('off')

# Grafiği Gösterin
plt.suptitle("Normalleştirilmiş MNIST Görüntüleri ve Etiketleri", fontsize=12)
plt.tight_layout()
plt.show()
```

This code selects and visualizes several sample images along with their labels from the MNIST dataset after normalization. Firstly, it imports the required libraries and loads the MNIST dataset. Then, it normalizes the images and selects the first 25 examples. These selected images are visualized in a figure containing a 5x5 grid of subplots, with each subplot displaying an image and its corresponding label.




```
# Import necessary libraries
```

```
import matplotlib.pyplot as plt
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Flatten, Activation
from keras.optimizers import Adam

# Load MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Preprocess data
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
train_images = train_images.astype('float32')
test_images = test_images.astype('float32')
train_images /= 255
test_images /= 255

# Create the neural network model
model = Sequential()

# Flatten the 28x28 input images into 784-dimensional vectors
model.add(Flatten(input_shape=(28, 28)))

# Add a hidden layer with 64 neurons and ReLU activation function
model.add(Dense(64, activation='relu'))

# Add an output layer with 10 neurons and softmax activation function for
classification
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=Adam(learning_rate=0.001),
              metrics=['accuracy'])

# Train the model on the training data
history = model.fit(train_images, train_labels, epochs=5, batch_size=32,
validation_data=(test_images, test_labels))

# Generate Training and Test Loss/Accuracy Plots
# Access training history using history.history dictionary
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```python
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Visualize Example Predictions
random_indices = np.random.randint(0, len(test_images), 10)
selected_images = test_images[random_indices]
selected_labels = test_labels[random_indices]

predictions = model.predict(selected_images)

for i, (image, label, pred) in enumerate(zip(selected_images, selected_labels,
predictions)):
    plt.subplot(5, 2, i + 1)
    plt.imshow(image, cmap='gray')
    plt.title(f'Label: {label}, Prediction: {np.argmax(pred)}')
    plt.axis('off')
plt.show()

# Evaluate the model's performance on the test data
score = model.evaluate(test_images, test_labels)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

This code constructs, trains, and evaluates a Convolutional Neural Network (CNN) model on the MNIST dataset. Firstly, it imports necessary libraries and loads the dataset. The data is preprocessed, and the model is defined, consisting of a convolutional layer, a flattening layer, a hidden layer, and an output layer. The model is compiled, trained on the training data, and its performance is evaluated on the test data. Additionally, example predictions are visualized.

```python

# Evaluate the model's performance on the test data
score = model.evaluate(test_images, test_labels)
print("Test loss:", score[0])
print("Test accuracy:", score[1])

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 655us/step - accuracy: 0.9754 - loss: 0.0843 - val_accuracy: 0.9702 - val_loss: 0.1010
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 715us/step - accuracy: 0.9792 - loss: 0.0714 - val_accuracy: 0.9734 - val_loss: 0.0928
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - accuracy: 0.9717 - loss: 0.1064
Test loss: 0.09284473210573196
Test accuracy: 0.9733999967575073
```

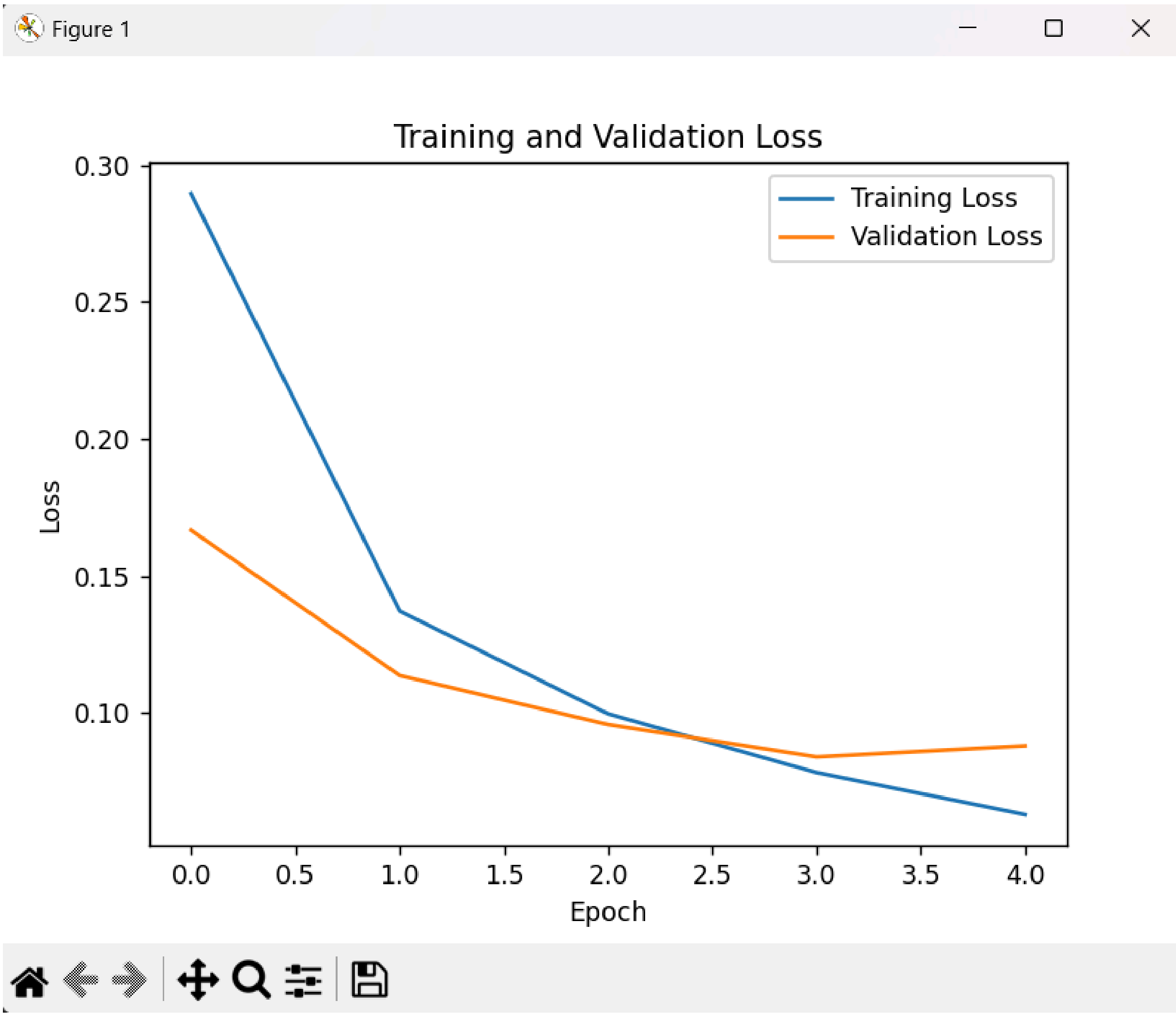


Figure 1
Training and Validation Loss
Training Loss
Validation Loss
0.30
0.25
0.20
0.15
0.10
Loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

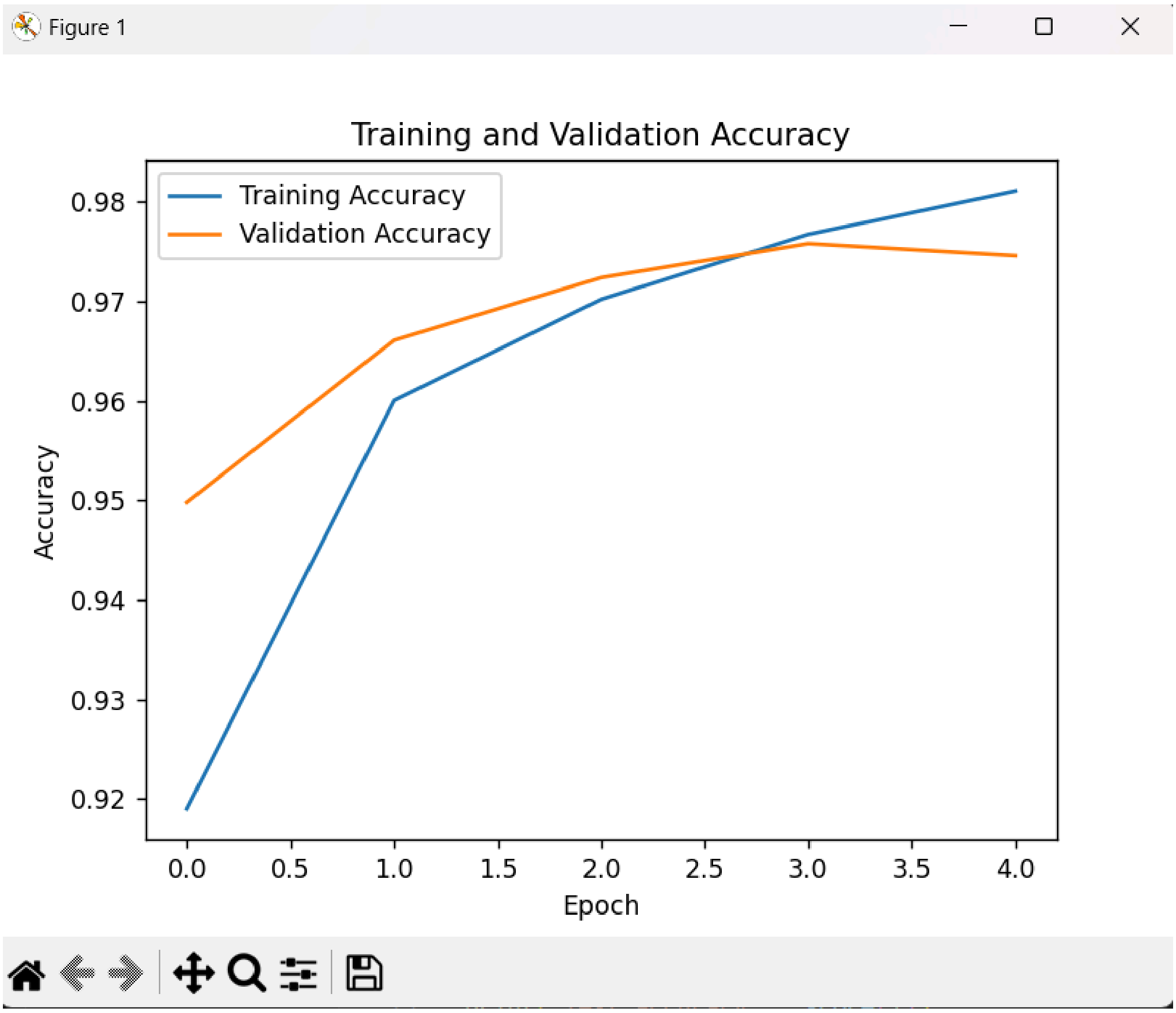


Figure 1
Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

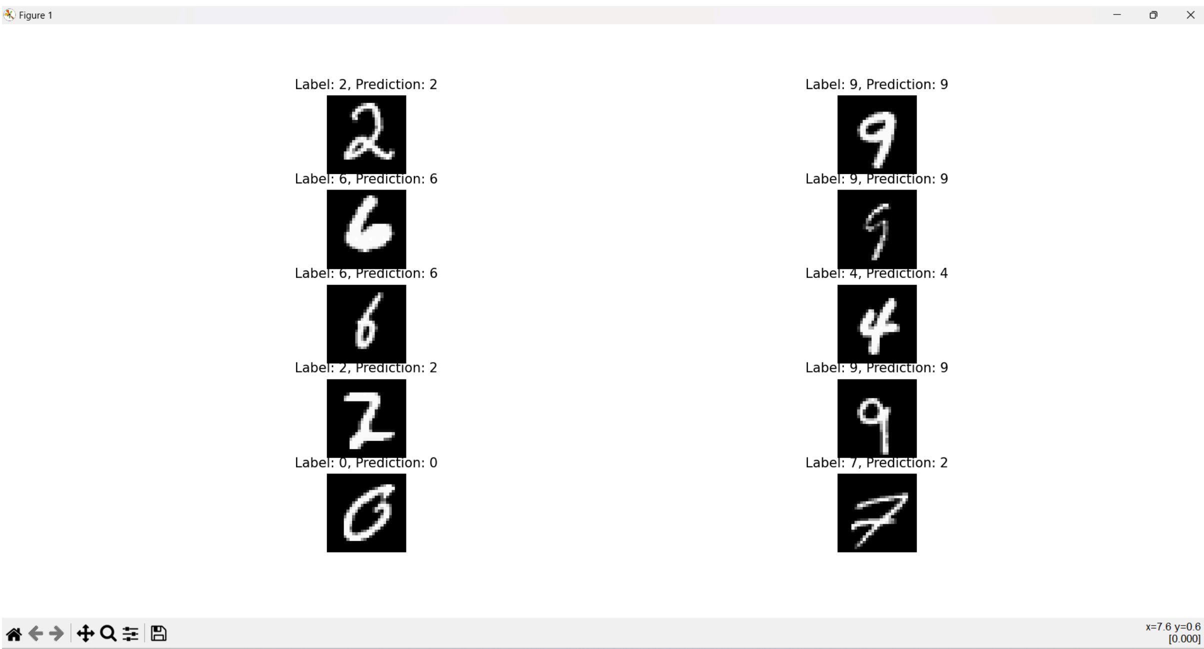
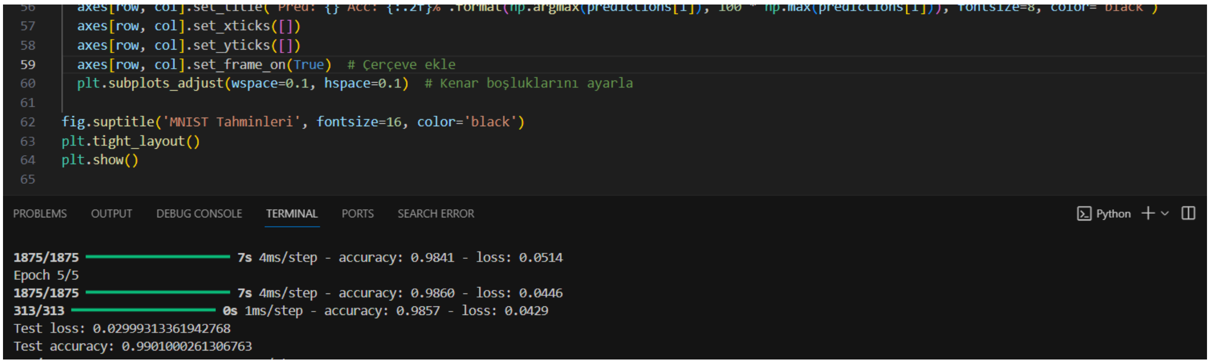


Figure 1
Label: 2, Prediction: 2
Label: 9, Prediction: 9
Label: 6, Prediction: 6
Label: 9, Prediction: 9
Label: 6, Prediction: 6
Label: 4, Prediction: 4
Label: 2, Prediction: 2
Label: 9, Prediction: 9
Label: 0, Prediction: 0
Label: 7, Prediction: 2

```
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical

# Verileri yükle
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Verileri ön işleme
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

# Hedef etiketleri one-hot encoding ile dönüştür
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10) # Test etiketlerine de one-hot
encoding uygulayın

# Model oluştur
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

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

# Modeli eğit
model.fit(X_train, y_train, epochs=5, batch_size=32)

# Modeli test et
score = model.evaluate(X_test, y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```python
# Tahminleri görselleştir
predictions = model.predict(X_test)

# Tek bir pencerede görselleştir
fig, axes = plt.subplots(3, 5, figsize=(15, 10))
for i in range(15):
  row = i // 5
  col = i % 5
  axes[row, col].imshow(X_test[i], cmap='coolwarm')
  axes[row, col].set_title('Pred: {} Acc:
{:.2f}%'.format(np.argmax(predictions[i]), 100 * np.max(predictions[i])),
fontsize=8, color='black')
  axes[row, col].set_xticks([])
  axes[row, col].set_yticks([])
  axes[row, col].set_frame_on(True)  # Çerçeve ekle
  plt.subplots_adjust(wspace=0.1, hspace=0.1)  # Kenar boşluklarını ayarla

fig.suptitle('MNIST Tahminleri', fontsize=16, color='black')
plt.tight_layout()
plt.show()
```

This code builds, trains, evaluates, and visualizes predictions of a CNN model on the MNIST dataset. It loads the data, performs preprocessing steps, defines the model, compiles, trains, and evaluates it. Finally, it evaluates the model's performance on the test data and visualizes example predictions.



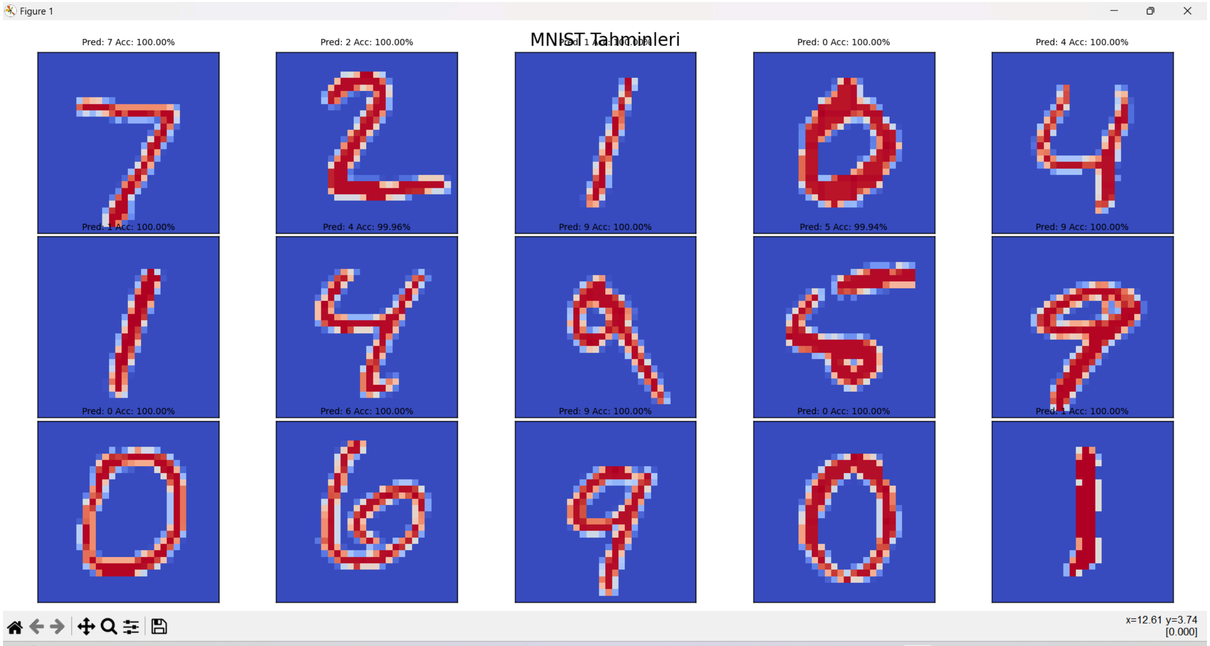


Figure 1
MNIST Tahminleri
Pred: 7 Acc: 100.00%
Pred: 2 Acc: 100.00%
Pred: 0 Acc: 100.00%
Pred: 4 Acc: 100.00%
Pred: 4 Acc: 99.96%
Pred: 9 Acc: 100.00%
Pred: 5 Acc: 99.94%
Pred: 9 Acc: 100.00%
Pred: 0 Acc: 100.00%
Pred: 6 Acc: 100.00%
Pred: 9 Acc: 100.00%
Pred: 0 Acc: 100.00%
x=12.61 y=3.74
[0.000]