T.C. KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ

YÜKSEK DÜZEY PROGRAMLAMA PROJE ÖDEVİ

DIGIT RECOGNIZER

BÜŞRA MERMER 202213171805

PROJENİN AMACI:

Bu projenin amacı, elle yazılmış rakamları (0-9) doğru bir şekilde sınıflandırabilen bir model oluşturmaktır. Bu, ünlü MNIST veri kümesi üzerinde eğitilmiş bir Evrişimli Sinir Ağı (CNN) kullanılarak elde edilir.

PROJEDE KULLANILAN METARYELLER:

1. Veri:

MNIST Veri Kümesi: Elle yazılmış rakam görüntülerinden oluşan ve makine öğrenmesi modellerini eğitmek için yaygın olarak kullanılan bir veri kümesidir. Projemde, bu veri kümesinin train.csv ve test.csv dosyaları olarak temsil edildiği görülmektedir. train.csv, modelin eğitilmesi için kullanılan etiketli verileri içerirken, test.csv modelin performansını değerlendirmek için kullanılan etiketsiz verileri içerir.

2. Yazılım ve Kütüphaneler:

- **Python:** Projenin temel programlama dilidir.
- **Pandas:** Verileri okumak, işlemek ve düzenlemek için kullanılan bir kütüphanedir.
- **NumPy:** Sayısal hesaplamalar yapmak ve dizilerle çalışmak için kullanılan bir kütüphanedir.
- **Matplotlib:** Verileri görselleştirmek ve grafikler oluşturmak için kullanılan bir kütüphanedir.
- **TensorFlow & Keras:** Derin öğrenme modelleri oluşturmak ve eğitmek için kullanılan kütüphanelerdir.
- **Scikit-learn:** Veri analizi ve makine öğrenmesi görevleri için çeşitli araçlar sağlayan bir kütüphanedir. Özellikle, projede veriyi eğitim ve doğrulama kümelerine ayırmak için train_test_split fonksiyonu kullandım.

İÇİNDEKİLER:

Veri Yükleme ve İşleme:

- **pd.read_csv():** Verileri CSV dosyalarından okumak ve Pandas DataFrame'lerine yüklemek için kullandım.
- **data.head():** Veri kümesinin ilk birkaç satırını görüntülemek ve veri yapısını anlamak için kullandım.
- **data.dtypes:** Veri kümesindeki sütunların veri türlerini kontrol etmek için kullandım.
- **data.shape:** Veri kümesinin boyutlarını (satır ve sütun sayısı) almak için kullandım.

Veri Ön İşleme:

- **train_test_split():** Veri kümesini eğitim ve doğrulama kümelerine ayırmak için kullandım.
- **to_categorical():** Etiketleri one-hot encoding formatına dönüştürmek için kullandım.
- **reshape():** Veri kümesinin boyutlarını değiştirmek ve CNN modeli için uygun hale getirmek için kullandım.
- **/255:** Piksel değerlerini 0 ile 1 arasında ölçeklendirmek için kullandım.

Model Oluşturma ve Eğitim:

- **Sequential():** Keras'ta sıralı bir model oluşturmak için kullandım.
- **Conv2D():** Evrişimli katmanlar eklemek için kullandım.
- **MaxPooling2D():** Havuzlama katmanları eklemek için kullandım.
- **Flatten():** Verileri düzleştirmek için kullandım.
- **Dense():** Tam bağlantılı katmanlar eklemek için kullandım.
- **compile():** Modeli derlemek ve kayıp fonksiyonunu, iyileştiriciyi ve metrikleri belirtmek için kullandım.
- **fit():** Modeli eğitmek için kullandım.
- **EarlyStopping():** Aşırı uyumu önlemek için erken durdurma mekanizmasını uygulamak için kullandım.

Model Değerlendirme ve Tahmin:

- **evaluate():** Modeli doğrulama verileri üzerinde değerlendirmek ve performans metriklerini elde etmek için kullandım.
- **predict():** Test verileri üzerinde tahminler yapmak için kullandım.
- **argmax():** Tahmin edilen olasılıkların en yüksek olduğu sınıfı bulmak için kullandım.
- **imshow():** Görüntüleri göstermek için kullandım.

KODLAR VE ÇIKTILARI:

Gerekli kütüphaneleri ve modülleri içe aktarma:

```
0
sn.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Input
from tensorflow.keras.callbacks import EarlyStopping

print('Kütüphaneler başarıyla yüklendi')
```

```
Kütüphaneler başarıyla yüklendi
```

Eğitim ve Test Verilerini CSV Dosyalarından Yükleme:

```
import pandas as pd

# Yüklenen dosyaları okuma
data_train = pd.read_csv('train.csv')
data_test = pd.read_csv('test.csv')
```

Eğitim Verilerini Özellikler (x) ve Etiketlere (y) Ayırma:

```
# Eğitim ve test verileri için NumPy dizisi oluştur
# Eğitim verilerinden etiket sütununu ayır ve etiketi y olarak al
x = data_train.loc[:, data_train.columns != 'label'].to_numpy()
y = data_train.loc[:, 'label'].to_numpy()
x_test = data_test.to_numpy()
# Sonuçları kontrol et
print(x[:5, :])
print('------------------------------------')
print(y[:5])
print('------------------------------------')
print(x_test[:5, :])
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
------------------------------------
[1 0 1 4 0]
------------------------------------
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Eğitim ve Doğrulama Kümelerine Bölme:

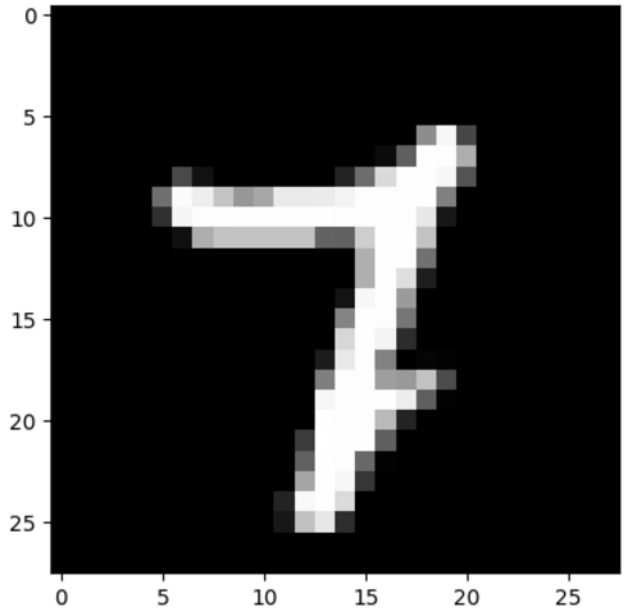
```
[508] x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.25, random_state=42)
      print(x_train.shape, y_train.shape, x_val.shape, y_val.shape)
```

```
(31500, 784) (31500,) (10500, 784) (10500,)
```

Görüntü ve Etiket Görüntüleme:

```
[510]
0
sn.
i = 0
print(y_train[i])
plt.imshow(x_train[i].reshape(28, 28), cmap='gray')
```

```
7
<matplotlib.image.AxesImage at 0x7aaa98034070>
```



Model Oluşturma:

```
# Modeli hazırlayın
model = Sequential()

# Input katmanını kullanarak giriş verilerinin şeklini tanımlayın, '1' gri tonlamalı resim anlamına gelir
model.add(Input(shape=(28, 28, 1)))

# Konvolüsyonel katman
model.add(Conv2D(filters=32, kernel_size=(4, 4), activation='relu'))

# Havuzlama (Pooling) katmanı
model.add(MaxPooling2D(pool_size=(2, 2)))

# Resimleri 28*28'den 764'e düzleştir (Flatten)
model.add(Flatten())

# 1. gizli katmanda nöron ekleyin
model.add(Dense(units=256, activation='relu'))

# 2. gizli katmanda nöron ekleyin
model.add(Dense(units=128, activation='relu'))

# 3. gizli katmanda nöron ekleyin
model.add(Dense(units=64, activation='relu'))

# Son katman: 10 olasılık sınıfı olan sınıflandırıcı
model.add(Dense(units=10, activation='softmax'))

# Kullanılabilir diğer metrikler için https://keras.io/metrics adresini kontrol edin
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Model Eğitimi:

```
model.fit(x=x_train,
          y=y_cat_train,
          validation_data=(x_val, y_cat_val),
          epochs=50,
          callbacks=[early_stop])
```

```
Epoch 1/50
985/985 ━━━━━━━━━━━━━━━━━━━━ 38s 37ms/step - accuracy: 0.8634 - loss: 0.4311 - val_accuracy: 0.9750 - val_loss: 0.0828
Epoch 2/50
985/985 ━━━━━━━━━━━━━━━━━━━━ 38s 39ms/step - accuracy: 0.9795 - loss: 0.0643 - val_accuracy: 0.9819 - val_loss: 0.0610
Epoch 3/50
985/985 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9861 - loss: 0.0405 - val_accuracy: 0.9828 - val_loss: 0.0538
Epoch 4/50
985/985 ━━━━━━━━━━━━━━━━━━━━ 38s 37ms/step - accuracy: 0.9929 - loss: 0.0221 - val_accuracy: 0.9790 - val_loss: 0.0706
Epoch 5/50
985/985 ━━━━━━━━━━━━━━━━━━━━ 41s 37ms/step - accuracy: 0.9938 - loss: 0.0198 - val_accuracy: 0.9790 - val_loss: 0.0749
<keras.src.callbacks.history.History at 0x7aaa95f75840>
```

Modelin Eğitim Geçmişini Düzenli Bir Tablo Formatına Dönüştürme:

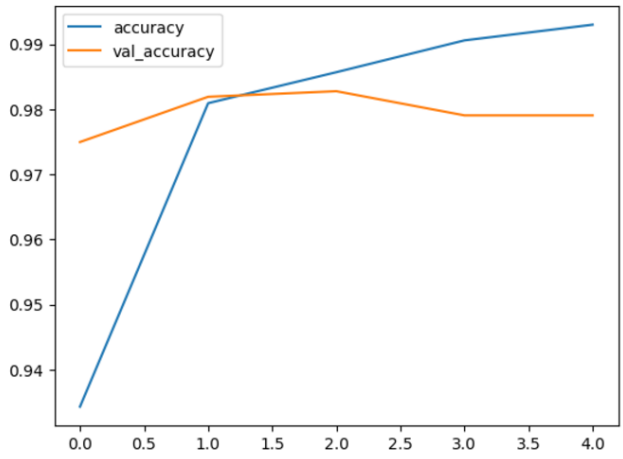
```
losses = pd.DataFrame(model.history.history)
losses
```

	accuracy	loss	val_accuracy	val_loss
0	0.934286	0.210516	0.974952	0.082849
1	0.980921	0.061251	0.981905	0.061047
2	0.985683	0.042692	0.982762	0.053781
3	0.990571	0.027805	0.979048	0.070563
4	0.992984	0.021959	0.979048	0.074921

Doğruluk Grafiği:

```
losses[['accuracy', 'val_accuracy']].plot()
```

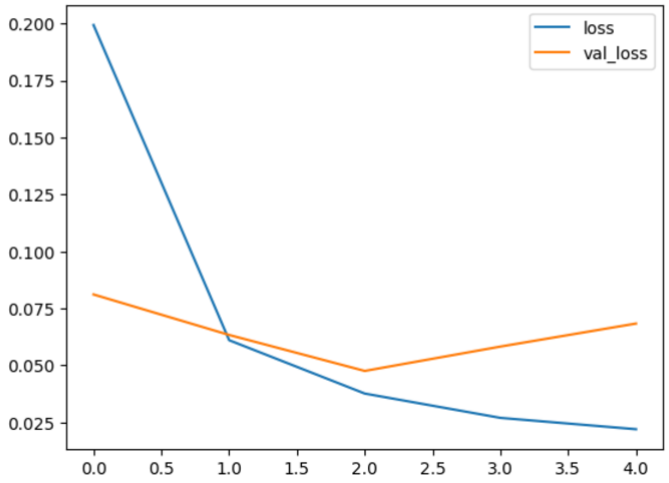
```
<Axes: >
```



Kayıp Grafiği:

```
losses[['loss', 'val_loss']].plot()
```

```
<Axes: >
```



Tahmin Yapma:

```
[530] predictions = [np.argmax(pred) for pred in model.predict(x_test)]
```

```
875/875 ━━━━━━━━━━━━━━━━━━━━ 9s 10ms/step
```

İlk Tahmini Gösterme:

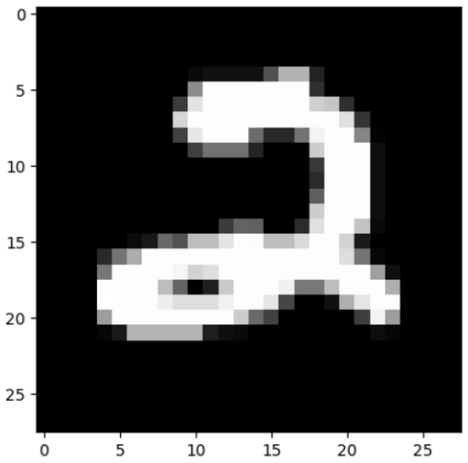
```
[535] predictions.__getitem__(0)# predictions listesinin ilk elemanını alır
```

```
2
```

İlk Test Görüntüsünü Gösterme:

```python
import matplotlib.pyplot as plt

# x_test[0] görüntüsünü siyah-beyaz (grayscale) olarak göster
plt.imshow(x_test[0], cmap='gray')
plt.show()
```



Test Verilerindeki Her Görüntü İçin Tahmin Edilen Rakam Değerini Kaydetme:

```python
# 'predictions' dizisini 'Label' adında yeni bir sütun olarak 'data_test' DataFrame'ine ekliyoruz
data_test['Label'] = predictions

# Güncellenmiş DataFrame'in ilk 5 satırını görüntülüyoruz
data_test.head()
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	Label
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	2
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	9
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	7
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	3

5 rows × 785 columns

Sonuçları Düzenleme:

```python
# data_test DataFrame'inden yalnızca 'Label' sütununu seç
test_df = data_test.loc[:, ['Label']]

# 1'den başlayarak 28000'e kadar sıralı bir 'ImageId' sütunu ekle
test_df['ImageId'] = range(1, len(data_test) + 1)

# 'ImageId' sütununu ilk sıraya alacak şekilde sütunları yeniden sırala
test_df = test_df[['ImageId', 'Label']]
```

Tahminleri CSV Dosyasına Kaydetme:

```
[551] test_df.to_csv('submission.csv', index=False)
```