YÜKSEK DÜZEY PROGRAMLAMA ÖDEVİ

PROJE RAPORU

Senanur Tunçbilek

202113172037

Digit Recognizer Veri Seti Eğitimi

1. Proje Tanımı

Bu projede, Kaggle platformunda sağlanan **Digit Recognizer** veri seti kullanılarak el yazısı rakamların sınıflandırılması üzerine bir yapay sinir ağı (CNN) modeli geliştirilmiştir. Model, 0-9 arası el yazısı rakamları sınıflandırmayı amaçlamaktadır. Proje boyunca aşağıdaki adımlar izlenmiştir:

1. Veri seti analizi ve ön işleme
2. Convolutional Neural Network (CNN) modeli geliştirme
3. Modelin eğitimi ve değerlendirilmesi
4. Test tahminlerinin gerçekleştirilmesi ve kaydedilmesi

---

2. Kullanılan Teknolojiler

- **Python**: Veri analizi ve model geliştirme
- **Kütüphaneler**:
    - **Pandas**: Veri işleme
    - **NumPy**: Matematiksel işlemler
    - **Matplotlib**: Görselleştirme
    - **TensorFlow/Keras**: Derin öğrenme modeli geliştirme
    - **Scikit-learn**: Veri bölme ve işleme

---

3. Veri Seti

3.1 Veri Setinin Tanımı

- **Kaynak**: Kaggle Digit Recognizer
- **Eğitim Verisi**:
    - Toplam 42,000 örnek
    - 28x28 boyutunda gri tonlamalı görüntüler
    - Etiketler: 0-9 arası rakam sınıfları

---

4. Adım Adım Proje Uygulaması

4.1 Veri Setinin Yüklenmesi

Kaggle üzerinden indirilen veri seti, Jupyter Notebook'ta Pandas kullanılarak yüklendi:

[3]:
```
# Gerekli kütüphaneleri yükleyelim
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D
from tensorflow.keras.optimizers import Adam
```

[5]:
```
# Veri setlerini yükleyelim
train_data = pd.read_csv('train.csv')  # Eğitim verisi
test_data = pd.read_csv('test.csv')    # Test verisi
```

Eğitim verisinin ilk birkaç satırı incelenmiştir:

```
[13]: # Eğitim verisinin ilk 5 satırını görüntüle
      print(train_data.head())

      # Test verisinin ilk 5 satırını görüntüle
      print(test_data.head())
```

```
   label  pixel0  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  \
0      1       0       0       0       0       0       0       0       0
1      0       0       0       0       0       0       0       0       0
2      1       0       0       0       0       0       0       0       0
3      4       0       0       0       0       0       0       0       0
4      0       0       0       0       0       0       0       0       0

   pixel8  ...  pixel774  pixel775  pixel776  pixel777  pixel778  pixel779  \
0       0  ...         0         0         0         0         0         0
1       0  ...         0         0         0         0         0         0
2       0  ...         0         0         0         0         0         0
3       0  ...         0         0         0         0         0         0
4       0  ...         0         0         0         0         0         0

   pixel780  pixel781  pixel782  pixel783
0         0         0         0         0
1         0         0         0         0
2         0         0         0         0
3         0         0         0         0
4         0         0         0         0
```

4.2 Veri Keşfi ve Ön İşleme

1. **Etiketlerin Ayrılması**:
    - label sütunu hedef değişken olarak ayrıldı:

```
[23]: # Etiketleri ve özellikleri ayır
      X = train_data.iloc[:, 1:].values  # Görüntü pikselleri
      y = train_data.iloc[:, 0].values  # Etiketler
```

2. **Normalizasyon**:
    - Piksel değerleri [0, 255] aralığından [0, 1] aralığına dönüştürüldü:

```
[25]: X = X / 255.0  # Piksel değerlerini 0-1 arasına getir
```

3. **Verinin Eğitim ve Doğrulama Olarak Bölünmesi**:S

- Eğitim verisi %80 eğitim ve %20 doğrulama olarak ikiye bölündü:

```
[27]: from sklearn.model_selection import train_test_split

# Eğitim ve doğrulama setlerine ayır
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

4. **Veri Yapısının CNN Modeline Uygun Hale Getirilmesi**:
   - 28x28x1 boyutuna yeniden şekillendirildi:

```
[29]: from tensorflow.keras.utils import to_categorical

# Etiketleri kategorik formata çevir
y_train = to_categorical(y_train, num_classes=10)
y_val = to_categorical(y_val, num_classes=10)
```

5. **Etiketlerin Kategorik Hale Getirilmesi**:


```
[29]: from tensorflow.keras.utils import to_categorical

# Etiketleri kategorik formata çevir
y_train = to_categorical(y_train, num_classes=10)
y_val = to_categorical(y_val, num_classes=10)
```


4.3 CNN Modelinin Geliştirilmesi

Modelin mimarisi aşağıdaki gibi tasarlanmıştır:

- **Conv2D ve MaxPooling Katmanları**: Görüntü özelliklerini çıkarmak için
- **Flatten Katmanı**: Veriyi 1D hale getirmek için
- **Dense Katmanları**: Tam bağlı katmanlarla sınıflandırma yapısı

[39]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# CNN modeli oluşturma
model = Sequential()

# İlk evrişim (convolution) katmanı
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))

# İkinci evrişim katmanı
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten katmanı
model.add(Flatten())

# Tam bağlı (dense) katman
model.add(Dense(128, activation='relu'))

# Çıkış katmanı
model.add(Dense(10, activation='softmax'))
```

Model derlenmiştir:

[43]:
```python
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Model özeti:

[45]:
```python
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_2 (Dense)	(None, 128)	204,928
dense_3 (Dense)	(None, 10)	1,290

Total params: 225,034 (879.04 KB)
Trainable params: 225,034 (879.04 KB)
Non-trainable params: 0 (0.00 B)

4.4 Modelin Eğitilmesi

Model 10 epoch boyunca eğitildi ve doğrulama verisiyle test edildi:

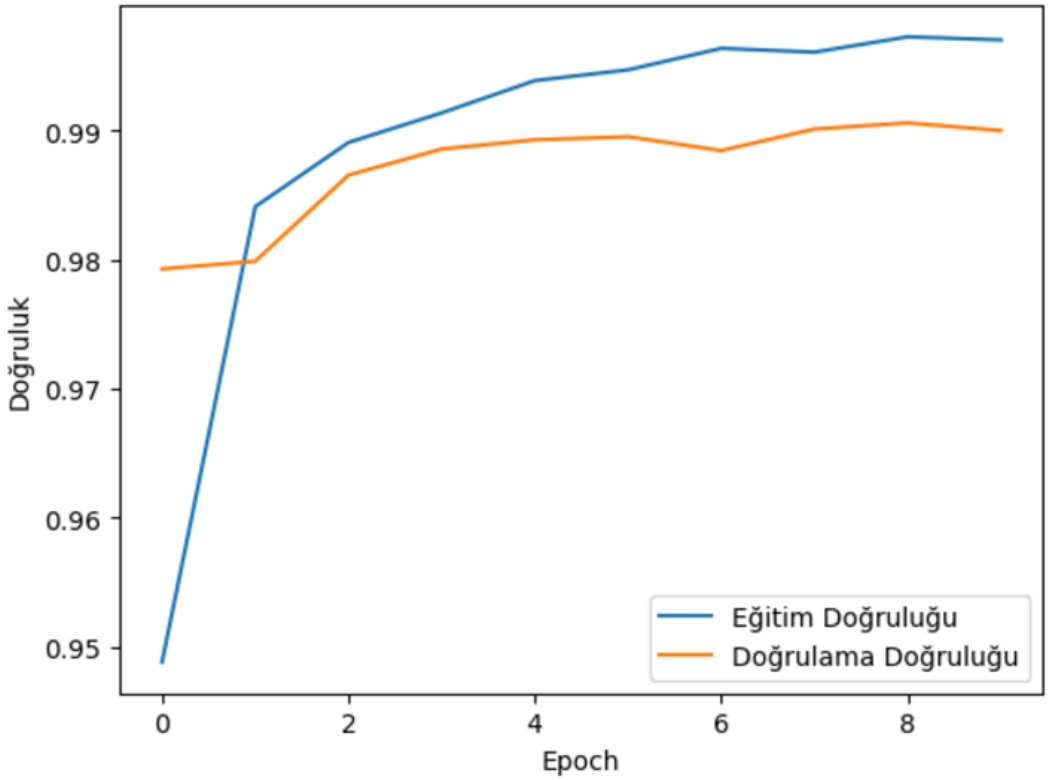
[47]:
```
# Modeli eğitme
history = model.fit(
    X_train, y_train,  # Eğitim verisi ve etiketleri
    validation_data=(X_val, y_val),  # Doğrulama verisi
    epochs=10,  # Eğitim süresi (daha fazla artırabilirsiniz)
    batch_size=32  # Mini-batch boyutu
)
```

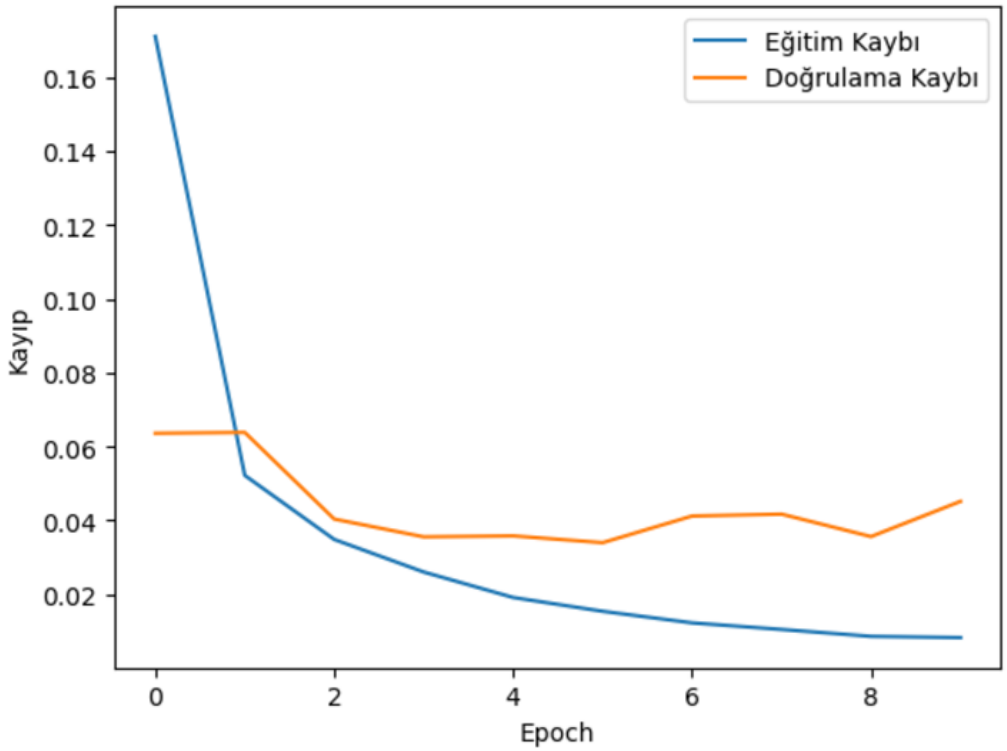
---

5. Sonuçlar

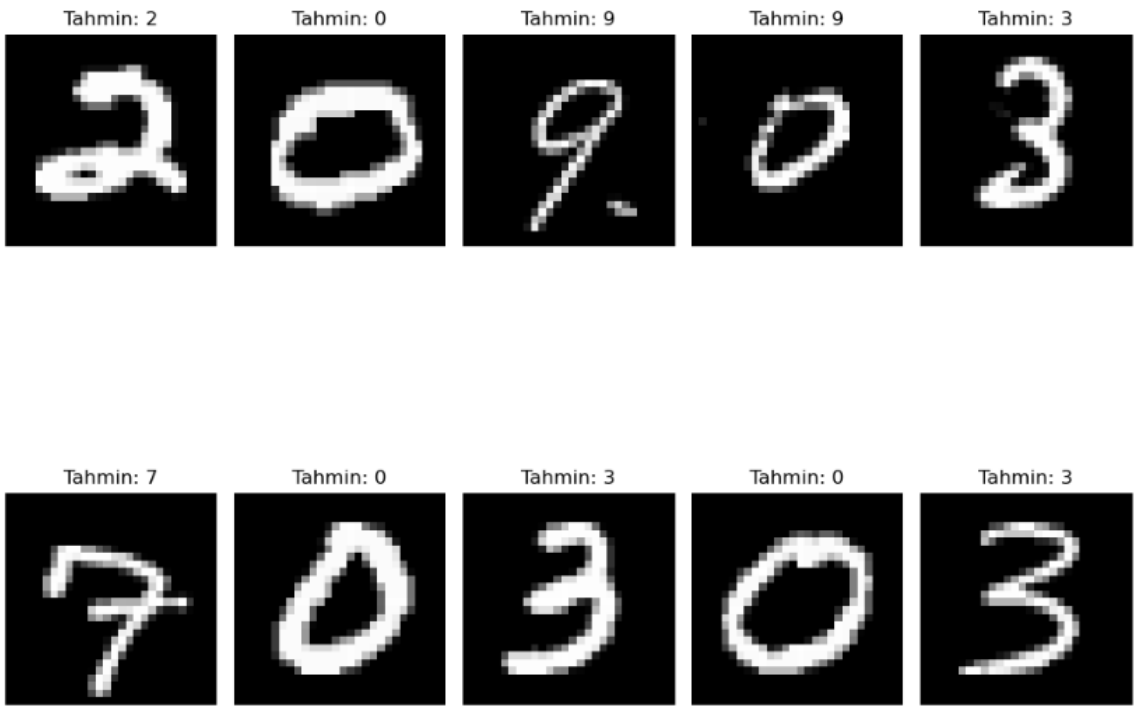
- **Eğitim Doğruluğu**: %99
- **Doğrulama Doğruluğu**: %99
- Modelin, test verisi üzerinde yüksek doğrulukta tahminler yaptığı görülmüştür.

[49]:
```
import matplotlib.pyplot as plt

# Eğitim ve doğrulama doğrulukları
plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')
plt.plot(history.history['val_accuracy'], label='Doğrulama Doğruluğu')
plt.xlabel('Epoch')
plt.ylabel('Doğruluk')
plt.legend()
plt.show()

# Eğitim ve doğrulama kayıpları
plt.plot(history.history['loss'], label='Eğitim Kaybı')
plt.plot(history.history['val_loss'], label='Doğrulama Kaybı')
plt.xlabel('Epoch')
plt.ylabel('Kayıp')
plt.legend()
plt.show()
```


Doğruluk
0.99
0.98
0.97
0.96
0.95
0
2
4
6
8
Epoch
Eğitim Doğruluğu
Doğrulama Doğruluğu


Kayıp
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0
2
4
6
8
Epoch
Eğitim Kaybı
Doğrulama Kaybı

Sonuç Çıktıları:



---

6. Çıktılar

- **Model Dosyası**: digit_recognizer_model.h5
- **Tahmin Dosyası**: submission.csv
- **Jupyter Notebook**: digit_recognizer_project.ipynb
