YÜKSEK DÜZEY PROGRAMLAMA DERSİ

Emirhan Büyüktaş
202013171035
Normal öğretim

Dicit recognizer

Amaç: Bu proje, el yazısı rakam görüntülerini otomatik olarak sınıflandırmak için bir derin öğrenme modeli eğitmeyi amaçlamaktadır. Bu amaçla, MNIST veri seti kullanılarak bir Evrişimli Sinir Ağı (CNN) modeli oluşturulmuş ve eğitilmiştir. Modelin performansı, test verileri üzerinde değerlendirilmiş ve elde edilen sonuçlar raporlanmıştır.

Veri Seti: Projede kullanılan MNIST veri seti, 0'dan 9'a kadar el yazısı rakam görüntülerinden oluşan büyük bir veri setidir. Veri seti, 60.000 eğitim görüntüsü ve 10.000 test görüntüsünden oluşmaktadır. Her görüntü 28x28 piksel boyutundadır ve gri tonlamalıdır.

Model: Projede kullanılan model, Evrişimli Sinir Ağı (CNN) modelidir. CNN'ler, görüntü işlemede yaygın olarak kullanılan bir derin öğrenme modeli türüdür. CNN'ler, görüntülerdeki özellikleri otomatik olarak öğrenme yetenekleri sayesinde el yazısı rakam tanıma gibi görevlerde başarılı sonuçlar elde etmektedir.

Kod Açıklamaları:

1. Kütüphanelerin İçe Aktarılması:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
```

Bu kısımda, projede kullanılacak olan kütüphaneler içe aktarılmaktadır. `numpy` sayısal işlemler, `pandas` veri işleme, `tensorflow` derin öğrenme, `matplotlib` grafik çizimi için kullanılmaktadır.

2. Kaggle Veri Setinin İndirilmesi:

```
!pip install -q kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
[8] !kaggle competitions download -c digit-recognizer
    !unzip digit-recognizer.zip
```

```
digit-recognizer.zip: Skipping, found more recently modified local copy (use --force to force download)
Archive:  digit-recognizer.zip
  inflating: sample_submission.csv
  inflating: test.csv
  inflating: train.csv
```

Bu kod bloğu, Kaggle platformundan MNIST veri setini indirmek için gerekli adımları içermektedir. Kaggle API'si kullanılarak veri seti indirilir ve ardından sıkıştırılmış dosya açılır.

```
[9] train_data = pd.read_csv('train.csv')
    test_data = pd.read_csv('test.csv')

    print("Eğitim verisi şekli:", train_data.shape)
    print("Test verisi şekli:", test_data.shape)
```

```
Eğitim verisi şekli: (42000, 785)
Test verisi şekli: (28000, 784)
```

- **`train.csv` ve `test.csv` dosyalarından eğitim ve test verileri sırasıyla `train_data` ve `test_data` DataFrame'lerine yüklenir.**
- **`train_data.shape` çıktısı (42000, 785), eğitim verilerinin 42000 örnek içerdiğini ve her örneğin 785 özellik (784 piksel değeri + 1 etiket) olduğunu gösterir.**
- **`test_data.shape` çıktısı (28000, 784), test verilerinin 28000 örnek içerdiğini ve her örneğin 784 özellik (piksel değerleri) olduğunu gösterir. Test verilerinde etiket bilgisi yoktur, çünkü modelin bu görüntüleri sınıflandırması beklenir.**

```
X = train_data.iloc[:, 1:].values.reshape(-1, 28, 28, 1) / 255.0  # Normalizasyon
y = to_categorical(train_data['label'], num_classes=10)
```

```
[12] X_test = test_data.values.reshape(-1, 28, 28, 1) / 255.0
```

```
[13] from sklearn.model_selection import train_test_split
     X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

Veri Hazırlama ve Bölme:

- **`X = train_data.iloc[:, 1:].values.reshape(-1, 28, 28, 1) / 255.0`: Eğitim verilerindeki piksel değerleri (ikinci sütundan itibaren) alınır, 28x28x1 boyutunda görüntüler olarak yeniden şekillendirilir ve 0-1 aralığına normalize edilir. Bu, modelin daha iyi performans göstermesine yardımcı olur. `X` değişkeni, eğitim verilerinin piksel değerlerini tutar.**
- **`y = to_categorical(train_data['label'], num_classes=10)`: Eğitim verilerindeki etiketler (`label` sütunu), 10 sınıflı (0-9 rakamları) kategorik verilere dönüştürülür. Bu, modelin çok sınıflı sınıflandırma**

yapabilmesi için gereklidir. y değişkeni, eğitim verilerinin etiketlerini tutar.

- `X_test = test_data.values.reshape(-1, 28, 28, 1) / 255.0`: Test verilerindeki piksel değerleri alınır, 28x28x1 boyutunda görüntüler olarak yeniden şekillendirilir ve 0-1 aralığına normalize edilir. `X_test` değişkeni, test verilerinin piksel değerlerini tutar.
- `from sklearn.model_selection import train_test_split X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)`: Eğitim verileri (X ve y), eğitim ve doğrulama setleri olarak ayrılır. `test_size=0.2`, verilerin %20'sinin doğrulama için kullanılacağı anlamına gelir. `random_state=42`, bölme işleminin tekrarlanabilir olmasını sağlar. Bu, modelin performansını daha doğru bir şekilde değerlendirmek için önemlidir.

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')  ])

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

Model Oluşturma, Derleme ve Özet

Bu kod parçası, Evrişimli Sinir Ağı (CNN) modelinin oluşturulması, derlenmesi ve özetinin görüntülenmesini içerir.

Model Oluşturma:

- `Sequential` modeli kullanılarak katmanlar sırayla eklenir:
  - İki adet `Conv2D` katmanı, görüntülerdeki özellikleri çıkarmak için kullanılır. 3x3 boyutunda filtreler ve ReLU aktivasyon fonksiyonu kullanılır.
  - İki adet `MaxPooling2D` katmanı, özellik haritalarının boyutunu azaltmak ve hesaplama maliyetini düşürmek için kullanılır.
  - `Flatten` katmanı, çok boyutlu verileri tek boyutlu bir vektöre dönüştürür.

- İki adet `Dense` katmanı, tam bağlantılı katmanlardır. İlk katmanda 128 nöron ve ReLU aktivasyonu, ikinci katmanda ise 10 nöron ve softmax aktivasyonu kullanılır. Softmax, çıktıları olasılıklara dönüştürerek sınıflandırma yapar.
- `Dropout` katmanı, aşırı öğrenmeyi önlemek için rastgele nöronları devre dışı bırakır.

Model Derleme:

- `model.compile` fonksiyonu ile model derlenir:
  - `optimizer='adam'`: Adam optimizasyon algoritması kullanılır.
  - `loss='categorical_crossentropy'`: Çok sınıflı sınıflandırma için kategorik çapraz entropi kayıp fonksiyonu kullanılır.
  - `metrics=['accuracy']`: Modelin performansını değerlendirmek için doğruluk metriği kullanılır.

Model Özeti:

- `model.summary()` fonksiyonu, modelin katmanlarını, çıktı şekillerini ve parametre sayısını özetleyen bir tablo görüntüler. Bu, modelin yapısını anlamak için faydalıdır.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_4 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_5 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_5 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_2 (Flatten)	(None, 1600)	0
dense_2 (Dense)	(None, 128)	204,928
dropout (Dropout)	(None, 128)	0
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)

```
[20] history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=3, batch_size=32)
```

```
Epoch 1/3
1050/1050 ━━━━━━━━━━━━━━━━━━━━ 36s 34ms/step - accuracy: 0.9719 - loss: 0.0883 - val_accuracy: 0.9864 - val_loss: 0.0433
Epoch 2/3
1050/1050 ━━━━━━━━━━━━━━━━━━━━ 38s 36ms/step - accuracy: 0.9817 - loss: 0.0635 - val_accuracy: 0.9875 - val_loss: 0.0399
Epoch 3/3
1050/1050 ━━━━━━━━━━━━━━━━━━━━ 38s 33ms/step - accuracy: 0.9855 - loss: 0.0458 - val_accuracy: 0.9867 - val_loss: 0.0437
```

```
[30] plt.figure(figsize=(6, 8))
     plt.subplot(2, 1, 2)
     plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')
     plt.plot(history.history['val_accuracy'], label='Doğrulama Doğruluğu')
     plt.legend()
     plt.title('Doğruluk Grafiği')

     plt.show()
```

Bu kod parçası, derlenen modelin eğitim verileri kullanılarak eğitilmesini sağlar.

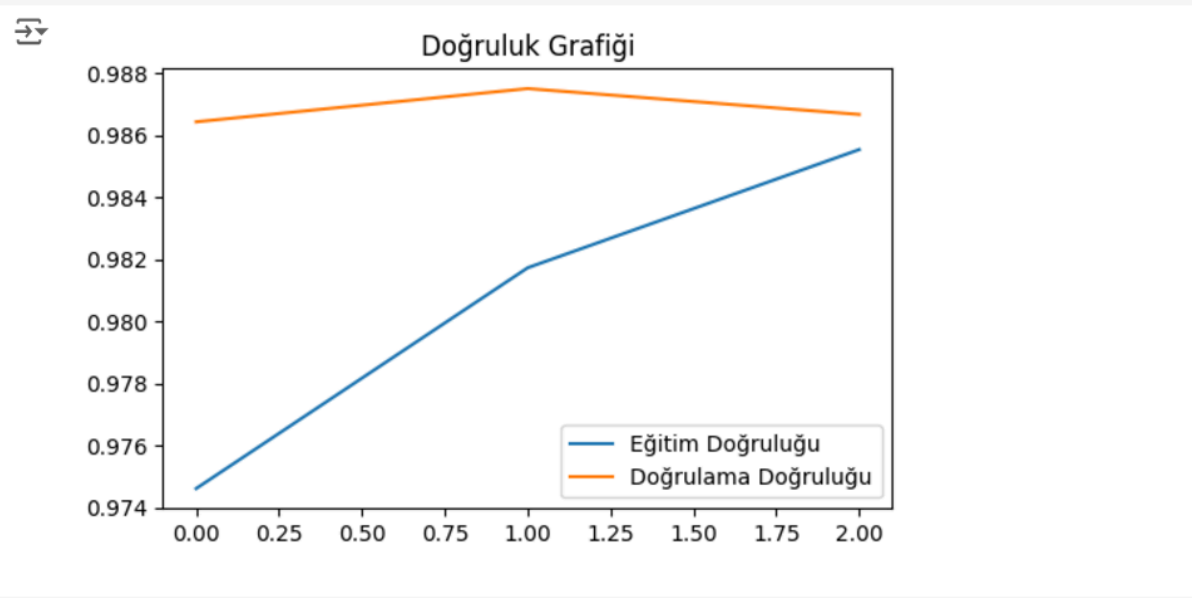
`history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=3, batch_size=32)`

- **`X_train, y_train`: Modelin eğitilmesi için kullanılan eğitim verileri ve etiketleri.**
- **`validation_data=(X_val, y_val)`: Modelin her epoch sonunda doğrulanması için kullanılan doğrulama verileri ve etiketleri. Bu, modelin aşırı öğrenmesini önlemeye yardımcı olur.**
- **`epochs=3`: Modelin tüm eğitim verileri üzerinde 3 kez eğitileceği anlamına gelir.**
- **`batch_size=32`: Her eğitim adımında modelin 32 örnek üzerinde eğitileceği anlamına gelir.**

`history`: Eğitim süreci boyunca kaydedilen metrikleri (doğruluk, kayıp vb.) içeren bir nesne. Bu bilgiler, modelin performansını değerlendirmek ve görselleştirmek için kullanılabilir.

Özetle, bu kod parçası, belirtilen parametreler kullanılarak modelin eğitim verileri üzerinde eğitilmesini ve eğitim süreci boyunca kaydedilen metriklerin `history` nesnesine kaydedilmesini sağlar. Bu adım, derin öğrenme modelinin performansını optimize etmek için gereklidir

Model Performansının Görselleştirilmesi: Doğruluk Grafiği



Bu kod parçası, modelin eğitim ve doğrulama sürecindeki doğruluk değerlerini görselleştirmek için bir grafik oluşturur.

`plt.figure(figsize=(6, 8))`: 6x8 inç boyutlarında bir grafik alanı oluşturur.

`plt.subplot(2, 1, 2)`: Grafik alanını 2 satır ve 1 sütunluk bir alt grafiğe böler ve ikinci alt grafiği seçer.

`plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')`: Eğitim verileri üzerindeki doğruluk değerlerini çizgi grafiği olarak çizer ve etiketi "Eğitim Doğruluğu" olarak ayarlar.

`plt.plot(history.history['val_accuracy'], label='Doğrulama Doğruluğu')`: Doğrulama verileri üzerindeki doğruluk değerlerini çizgi grafiği olarak çizer ve etiketi "Doğrulama Doğruluğu" olarak ayarlar.

`plt.legend()`: Grafiğe bir açıklama ekler, böylece çizgilerin hangi veriyi temsil ettiği anlaşılır.

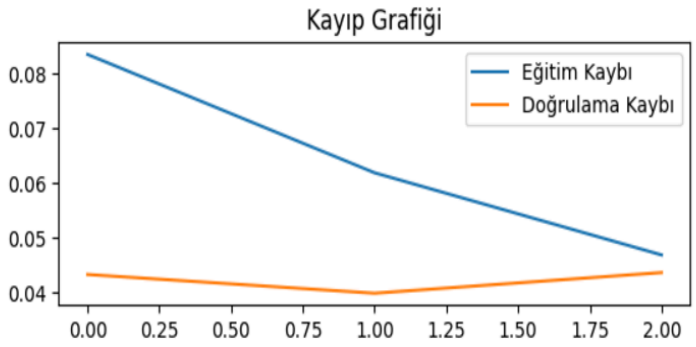
`plt.title('Doğruluk Grafiği')`: Grafiğe "Doğruluk Grafiği" başlığını ekler.

`plt.show()`: Oluşturulan grafiği görüntüler.

Özetle, bu kod parçası, modelin eğitim ve doğrulama sürecindeki doğruluk değerlerini görselleştirerek, modelin performansını analiz etmeyi sağlar. Grafik, eğitim ve doğrulama doğruluklarının epoch'lara göre nasıl değiştiğini gösterir. Bu, modelin öğrenme sürecini ve aşırı öğrenme olup olmadığını anlamak için önemlidir.

Model Performansının Görselleştirilmesi: Kayıp Grafiği

```
plt.subplot(2, 1, 2)
plt.plot(history.history['loss'], label='Eğitim Kaybı')
plt.plot(history.history['val_loss'], label='Doğrulama Kaybı')
plt.legend()
plt.title('Kayıp Grafiği')
plt.show()
```



Bu kod parçası, modelin eğitim ve doğrulama sürecindeki kayıp (loss) değerlerini görselleştirmek için bir grafik oluşturur.

`plt.subplot(2, 1, 2)`: Grafik alanını 2 satır ve 1 sütunluk bir alt grafiğe böler ve ikinci alt grafiği seçer. (Muhtemelen önceki kod bloğundan devam ediyor, bu satıra gerek duyulmayabilir.)

`plt.plot(history.history['loss'], label='Eğitim Kaybı')`: Eğitim verileri üzerindeki kayıp değerlerini çizgi grafiği olarak çizer ve etiketi "Eğitim Kaybı" olarak ayarlar.

`plt.plot(history.history['val_loss'], label='Doğrulama Kaybı')`: Doğrulama verileri üzerindeki kayıp değerlerini çizgi grafiği olarak çizer ve etiketi "Doğrulama Kaybı" olarak ayarlar.

`plt.legend()`: Grafiğe bir açıklama ekler, böylece çizgilerin hangi veriyi temsil ettiği anlaşılır.

`plt.title('Kayıp Grafiği')`: Grafiğe "Kayıp Grafiği" başlığını ekler.

`plt.show()`: Oluşturulan grafiği görüntüler.

Özetle, bu kod parçası, modelin eğitim ve doğrulama sürecindeki kayıp değerlerini görselleştirerek, modelin öğrenme sürecini analiz etmeyi sağlar. Grafik, eğitim ve doğrulama kayıplarının epoch'lara göre nasıl değiştiğini gösterir. Kayıp değerlerinin

azalması, modelin öğrenme sürecinde ilerlediğini gösterir. Eğitim ve doğrulama kayıpları arasındaki fark, modelin aşırı öğrenme eğilimini gösterir.

Model Tahminlerinin Görselleştirilmesi

```python
import numpy as np

# Test seti üzerinde tahmin yap
predictions = model.predict(X_test)

# Rastgele bir örneği seç ve sonucu gör
index = np.random.randint(0, len(X_test))
plt.imshow(X_test[index].reshape(28, 28), cmap='gray')



# Seçenek 1: Yalnızca tahmini göster
plt.title(f"Tahmin: {np.argmax(predictions[index])}")

# Seçenek 2: Karşılaştırma için doğrulama setini kullan
index = np.random.randint(0, len(X_val))
plt.imshow(X_val[index].reshape(28, 28), cmap='gray')
plt.title(f"Gerçek: {np.argmax(y_val[index])}, Tahmin: {np.argmax(model.predict(X_val[index].reshape(1, 28, 28, 1))[0])}")

plt.show()
```

Bu kod parçası, eğitilen modelin test ve doğrulama verileri üzerindeki tahminlerini görselleştirerek, modelin performansını örnekler üzerinde incelemeyi sağlar.

`predictions = model.predict(X_test)`: Eğitilen model kullanılarak test verileri üzerinde tahminler yapılır ve `predictions` değişkenine kaydedilir.

`index = np.random.randint(0, len(X_test))`: Test verileri arasından rastgele bir örnek seçilir.

`plt.imshow(X_test[index].reshape(28, 28), cmap='gray')`: Seçilen örneğin görüntüsü gri tonlamalı olarak gösterilir.

`plt.title(f"Tahmin: {np.argmax(predictions[index])}")`: Modelin seçilen örnek için yaptığı tahmin (0-9 arası bir rakam) görüntünün üzerine başlık olarak yazdırılır.

`index = np.random.randint(0, len(X_val))`: **Doğrulama verileri arasından rastgele bir örnek seçilir.**

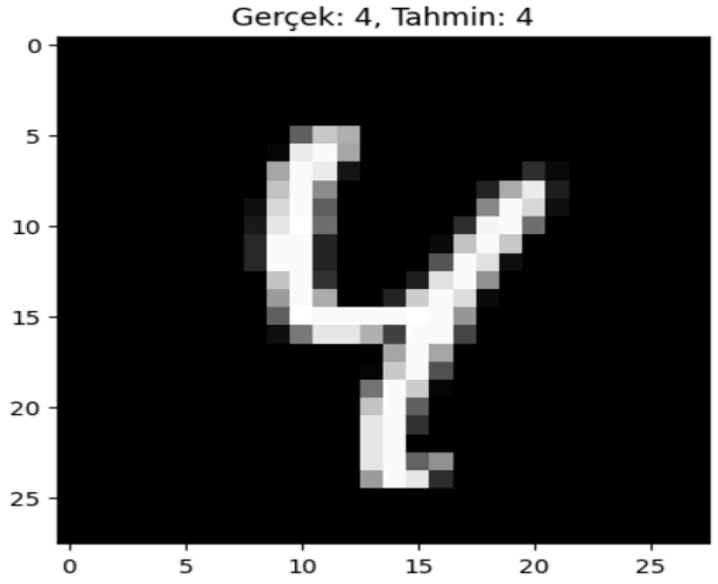
`plt.imshow(X_val[index].reshape(28, 28), cmap='gray')`: **Seçilen doğrulama örneğinin görüntüsü gri tonlamalı olarak gösterilir.**

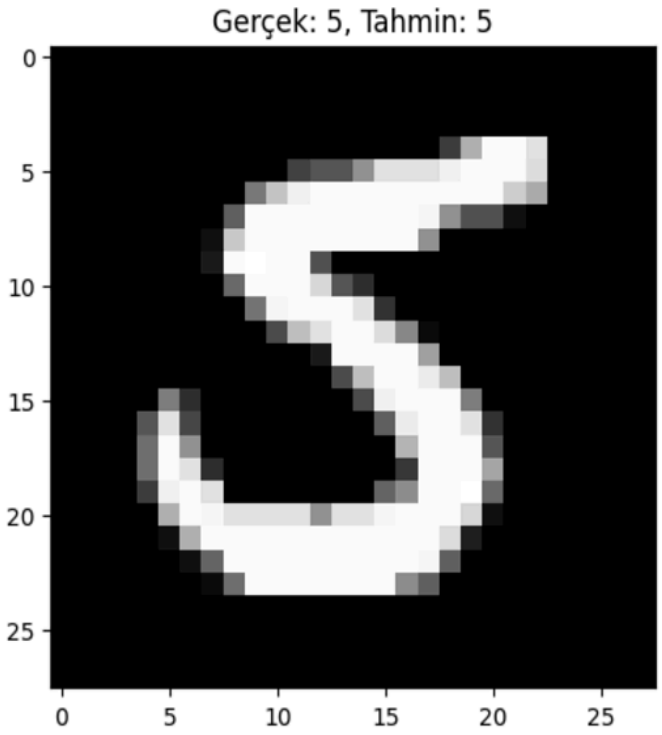
`plt.title(f"Gerçek: {np.argmax(y_val[index])}, Tahmin: {np.argmax(model.predict(X_val[index].reshape(1, 28, 28, 1))[0])}")`: **Doğrulama örneği için gerçek etiket ve modelin tahmini görüntünün üzerine başlık olarak yazdırılır. Bu, modelin performansını doğrudan karşılaştırmayı sağlar.**

`plt.show()`: **Oluşturulan görüntüler ekranda gösterilir.**

Özetle, bu kod parçası, modelin test ve doğrulama verileri üzerindeki tahminlerini görselleştirerek, modelin performansını örnekler üzerinde değerlendirmeyi sağlar. Rastgele seçilen örnekler için görüntüler ve tahminler gösterilir. Doğrulama verileri için gerçek etiket ve tahmin karşılaştırılarak modelin doğruluğu hakkında fikir edinilir.

```
875/875 ━━━━━━━━━━━━━━━━━━━━ 8s 10ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
```




Gerçek: 5, Tahmin: 5
0
5
10
15
20
25
0
5
10
15
20
25