
T.C Kütahya Dumlupınar Üniversitesi

Bilgisayar Mühendisliği Bölümü

Yüksek Düzey Programlama

- Digit Recognizer -

202213171071
ÖMER FARUK UYANIK

# PROJENİN AMACI

•Projeniz **El Yazısı Rakam Tanıma** üzerine odaklanıyor. Kısaca özetlemek gerekirse, yaptıklarınız şunlardır:

- **Veri Yükleme ve Hazırlık:** MNIST veri kümesini veya benzer bir el yazısı rakam veri kümesini kullanarak eğitim ve test verilerinizi yükledik. Verileri temizledik, piksel değerlerini normalleştirdik ve CNN modeli için uygun formata getirdik
- **Model Oluşturma ve Eğitim:** Sinir Ağı (CNN) kullanarak bir model mimarisi tasarladık. Bu model, el yazısı rakamları tanımak için eğitildi.
- **Model Değerlendirme:** Modelinizi doğrulama verileri üzerinde değerlendirdiniz ve doğruluğunu ölçtük. Eğitim ve doğrulama kayıplarını izleyerek modelin performansını inceledik.
- **Tahmin ve Sonuç:** Eğitilmiş modeli kullanarak test verileri üzerinde tahminler yaptınız ve sonuçları "rakam_tanima_sonuc.csv" adlı bir dosyaya kaydettik. Bu dosya, her bir test görüntüsü için tahmin edilen rakam etiketini içerir.
- **Görselleştirme:** Eğitim ve doğrulama doğruluklarını, kayıplarını ve modelin tahminlerini görselleştirmek için grafikler kullandık. Bu, modelin davranışını ve performansını daha iyi anlamanıza yardımcı oldu.

# EL YAZISI RAKAM TANIMA PROJESİNDE KULLANILAN MATARYEL VE METOTLAR

## •Materyaller:

- **Veri Kümesi:** Model eğitimi ve testi için bir el yazısı rakam veri kümesi kullandım. **MNIST** veri kümesiydi, çünkü el yazısı rakam tanıma alanında yaygın olarak kullanılan standart bir veri kümesidir. MNIST, 0'dan 9'a kadar olan rakamların 28x28 piksel boyutunda gri tonlamalı görüntülerini içerir.
- **Donanım:** Projemi Google Colab ortamında geliştirdiğim için Google'ın sağladığı sunucuları ve GPU'ları kullandım. Bu sayede modeli eğitmek için gereken hesaplama gücüne erişebildim.
- **Yazılım:** Projemi Python programlama diliyle yazdım ve çeşitli kütüphanelerden yararlandım. Bunlar arasında:
  - **TensorFlow/Keras:** Derin öğrenme modelleri oluşturmak ve eğitmek için kullanılan popüler kütüphaneler.
  - **Pandas:** Veri işleme ve analiz için kullandığım kütüphane.
  - **NumPy:** Sayısal hesaplamalar için olmazsa olmaz bir kütüphane.
  - **Scikit-learn:** Makine öğrenmesi algoritmaları ve araçları için faydalandığım kütüphane.
  - **Matplotlib/Seaborn:** Veri görselleştirme amacıyla kullandığım kütüphaneler.

## Metotlar:

- **Veri Ön İşleme:** Modeli eğitmeden önce ham verileri modele uygun hale getirmek için ön işleme adımlarını uyguladım. Bunlar:
- **Veri Temizleme:** Eksik veya hatalı verileri düzelttim.
- **Normalleştirme:** Piksel değerlerini 0 ile 1 arasında ölçeklendirdim.
- **Yeniden Şekillendirme:** Görüntüleri CNN modelinin beklediği formata dönüştürdüm.
- **Model Oluşturma:** El yazısı rakamları tanımak için **Evrişimli Sinir Ağı (CNN)** modelini tercih ettim. CNN'ler, görüntü işlemede oldukça başarılı sonuçlar veren bir derin öğrenme modeli türüdür. Modelim, evrişim katmanları, pooling katmanları, düzleştirme katmanı ve tam bağlı katmanlardan oluşuyor.
- **Model Eğitimi:** Eğitim verilerini kullanarak modeli eğittim. Bu süreçte, modelin ağırlıkları, eğitim verilerindeki örüntüleri öğrenecek şekilde ayarlandı.
- **Model Değerlendirme:** Eğitilmiş modelin performansını daha önce görmediği veriler üzerinde ölçmek için doğrulama verilerini kullandım. Modelin performansını değerlendirmek için temel metrik olarak **doğruluk** kullandım.
- **Hiperparametre Ayarlama:** Modelin performansını optimize etmek için hiperparametrelerle oynadım. Farklı hiperparametre değerlerini deneyerek en iyi sonuçları verenleri seçtim.
- **Tahmin:** Eğitilmiş modeli kullanarak test verileri üzerinde tahminler yaptım. Bu tahminler, el yazısı rakamların tanınmasını sağladı.
- **Sonuçların Görselleştirilmesi:** Modelin performansını ve tahminlerini daha iyi anlamak için grafikler ve görselleştirmeler oluşturdum.

# PROJEDEKİ ADIMLAR :
# PROJEDEKİ VERİ SETİ, MODEL EĞİTİMİ, NORMALİZASYON GİBİ AŞAMALARIN AÇIKLANMASI

•**Veri Yükleme:** `train.csv` ve `test.csv` dosyalarından verileri yükledik.

- **Veri İnceleme:** `head()`, `dtypes`, `shape`, `columns` gibi fonksiyonlar kullanarak verilerin yapısını ve özelliklerini inceledik.
- **Ön İşleme:**
  - Eğitim ve test verilerini NumPy dizilerine dönüştürdük.
  - Verileri eğitim (%75) ve doğrulama (%25) setlerine ayırdık (`train_test_split` kullanarak).
  - Etiketleri one-hot encoding formatına dönüştürdük (`to_categorical` kullanarak).
- **Normalizasyon:**
  - Piksel değerlerini 0 ile 1 arasına ölçeklendirdik (255'e bölerek).
- **Veri Şekillendirme:**
  - Eğitim, doğrulama ve test verilerini CNN için uygun 4 boyutlu hale getirdik (`reshape` kullanarak).
- **Model Oluşturma ve Eğitim:**
  - TensorFlow ve Keras kullanarak bir CNN modeli oluşturduk.
  - Modeli eğitim verileriyle eğittiniz ve doğrulama verileriyle performansını değerlendirdik.
- **Tahmin ve Sonuç:**
  - Test verileri üzerinde tahmin yaptık.
  - Sonuçları `rakam_tanima_sonuc.csv` dosyasına kaydettik.
- **Değerlendirme:**
  - Eğitim ve doğrulama doğruluğunu görselleştirdim.
  - Eğitim ve doğrulama kayıplarını çizdirdik.
  - Rastgele örnekler üzerinde tahminleri görselleştirdik.

•**Veri Bölünmesi:**

- Verilerinizi `train_test_split` fonksiyonu kullanarak %75 eğitim ve %25 doğrulama seti olarak ayırdık. `test_size=0.25` parametresiyle doğrulama setinin boyutunu belirledik.
- **Normalizasyon:**
- Verilerinizi 0 ile 1 arasına ölçeklendirmek için piksel değerlerini 255'e böldük. Bu işlem, modelin daha hızlı ve daha iyi öğrenmesine yardımcı olur.

- **Özetle:**
- Projenizde, el yazısı rakamları tanımak için bir CNN modeli oluşturduk. Verileri ön işledik, normalleştirdik, modeli eğittik ve test verileri üzerinde tahminler yaptık. Sonuçları bir dosyaya kaydederek ve performansı görselleştirerek değerlendirdik.

# KULLANILAN MODEL HAKKINDA BİLGİ VE DOĞRULUK ORANI

Kodumda, el yazısı rakamları tanımak için bir **Evrişimli Sinir Ağı (CNN)** modeli kullandım. CNN'ler, özellikle görüntü işleme görevlerinde başarılı sonuçlar veren derin öğrenme modelleridir.

- Modelin mimarisi şu şekildedir:
- **Giriş Katmanı:** 28x28 piksel boyutunda gri tonlamalı görüntüler alır.
- **Evrişim Katmanları:** Görüntüdeki özellikleri çıkarmak için kullanılır. İki evrişim katmanı bulunmaktadır; ilki 32 filtreli, ikincisi ise 64 filtrelidir. Her ikisi de ReLU aktivasyon fonksiyonunu kullanır.
- **Max-Pooling Katmanları:** Özellik haritalarının boyutunu küçültmek ve hesaplama yükünü azaltmak için kullanılır. İki max-pooling katmanı bulunmaktadır.
- **Düzleştirme Katmanı:** Evrişim katmanlarından elde edilen özellik haritasını, tam bağlı katmanlara giriş olarak kullanılabilecek tek boyutlu bir vektöre dönüştürür.
- **Tam Bağlı Katmanlar:** Sınıflandırma için kullanılır. İki tam bağlı katman bulunmaktadır; ilki 64 birimli ve ReLU aktivasyon fonksiyonunu kullanır, ikincisi ise 10 birimli ve softmax aktivasyon fonksiyonunu kullanır. Çıkış katmanı 10 birimlidir çünkü 10 farklı rakam (0-9) sınıflandırılmaktadır.
- Model, `adam` optimizasyon algoritması ve `sparse_categorical_crossentropy` kayıp fonksiyonu kullanılarak eğitilmiştir. Doğruluk metriği olarak ise `accuracy` kullanılmıştır.

- Doğruluk Oranı
- Modelin doğruluğunu değerlendirmek için doğrulama seti kullanılmıştır. Doğrulama doğruluğu, modelin daha önce görmediği veriler üzerindeki performansını ölçer. Kodumda, doğrulama doğruluğu `model.evaluate()` fonksiyonu kullanılarak hesaplanmış ve şu şekilde yazdırılmıştır:

- ```
  dogrulama_dogrulugu = model.evaluate(X_dogrulama, y_dogrulama, verbose=2)
  print(f'Doğrulama Doğruluğu: {dogrulama_dogrulugu[1]:.4f}')
  ```

- Bu kod, doğrulama setindeki doğruluk oranını hesaplar ve ekrana yazdırır. Örneğin, doğruluk oranı 0.98 ise, modelin doğrulama setindeki verilerin %98'ini doğru bir şekilde sınıflandırdığı anlamına gelir.
  ```

# PROJEDEKİ ÖNEMLİ KOD PARÇALARININ AÇIKLANMASI

**1. Veri Ön İşleme**

- Veri ön işleme aşaması, modelin eğitilmesi ve değerlendirilmesi için verilerin hazırlanması sürecidir. Projemde, aşağıdaki adımları içeren bir veri ön işleme süreci kullandım:

```
# Verileri 0 ile 1 Arasında Ölçeklendirme
x_egitim = x_egitim / 255.0
x_dogrulama = x_dogrulama / 255.0
test_verisi = test_verisi / 255.0

# Eğitim Setini 4 Boyutlu Hale Getirme
x_egitim = x_egitim.reshape(31500, 28, 28, 1)

# Doğrulama Setini 4 Boyutlu Hale Getirme
x_dogrulama = x_dogrulama.reshape(10500, 28, 28, 1)

# Test Setini 4 Boyutlu Hale Getirme
test_verisi = test_verisi.reshape(28000, 28, 28, 1)
```

**•Açıklama:**

- **Ölçeklendirme:** Piksel değerlerini 0 ile 1 arasında ölçeklendirdim. Bu işlem, modelin daha hızlı ve daha doğru bir şekilde öğrenmesine yardımcı olur.
- **Boyut Dönüşümü:** CNN modeli, 4 boyutlu girdi verisi (örnek sayısı, yükseklik, genişlik, kanal sayısı) bekler. Bu nedenle, eğitim, doğrulama ve test verilerini 4 boyutlu hale getirdim.

## 2. Model Oluşturma

- Projemde, bir Evrişimli Sinir Ağı (CNN) modeli kullandım. Modelin mimarisi şu şekildedir:
- ```
  model = models.Sequential([
      layers.Input(shape=(28, 28, 1)),
      layers.Conv2D(32, (3, 3), activation='relu'),
      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')
  ])
  ```

•**Açıklama:**

- **Sequential Model:** Keras'ın `Sequential` API'sini kullanarak modeli oluşturdum. Bu API, katmanları sırayla ekleyerek model oluşturmayı kolaylaştırır.
- **Evrişim Katmanları:** Görüntüdeki özellikleri çıkarmak için iki evrişim katmanı kullandım.
- **Max-Pooling Katmanları:** Özellik haritalarının boyutunu küçültmek ve hesaplama yükünü azaltmak için max-pooling katmanları ekledim.
- **Düzleştirme Katmanı:** Evrişim katmanlarından elde edilen özellik haritasını, tam bağlı katmanlara giriş olarak kullanılabilecek tek boyutlu bir vektöre dönüştürdüm.
- **Tam Bağlı Katmanlar:** Sınıflandırma için iki tam bağlı katman kullandım. Son katman, 10 farklı rakamı (0-9) temsil eden 10 birimli ve softmax aktivasyon fonksiyonlu bir çıkış katmanıdır.
  ```

## 3. Model Eğitimi

- Modeli eğitmek için `model.fit()` fonksiyonunu kullandım:
- ```
  gecmis = model.fit(
      X_egitim_ayrik, y_egitim_ayrik,
      epochs=10,
      validation_data=(X_dogrulama, y_dogrulama)
  )
  ```

•**Açıklama:**

- **Eğitim Verileri:** Modeli, eğitim verileri (`X_egitim_ayrik`, `y_egitim_ayrik`) ile eğittim.
- **Doğrulama Verileri:** Modelin performansını, doğrulama verileri (`X_dogrulama`, `y_dogrulama`) üzerinde izleyerek aşırı öğren

## •Proje Özeti: El Yazısı Rakam Tanıma

- Bu projede, **MNIST veri seti** kullanılarak el yazısı rakamları (0-9) tanıyan bir model oluşturmayı amaçladım.
- **Ön İşleme:** Veriler temizlendi, normalleştirildi (0-1 arası) ve CNN modeli için uygun formata (28x28 piksel, 1 kanal) getirildi.
- **Model: Evrişimli Sinir Ağı (CNN)** kullanarak bir model tasarladım. Modelde evrişim katmanları, max-pooling katmanları, düzleştirme katmanı ve tam bağlı katmanlar yer alıyor.
- **Eğitim:** Verinin %75'i eğitim, %25'i doğrulama için ayrıldı. Model, **Adam optimizer** ve **sparse categorical crossentropy** kayıp fonksiyonu ile eğitildi.
- **Değerlendirme:** Doğrulama seti üzerinde modelin doğruluğu ölçüldü.
- **Tahmin:** Eğitilen model, test verileri üzerinde tahminlerde bulundu ve sonuçlar "rakam_tanima_sonuc.csv" dosyasına kaydedildi.
- **Sonuç:** Proje, el yazısı rakamları yüksek doğrulukla tanıyan bir CNN modeli oluşturmayı başardı.
  ```