**Yüksek Düzey Programlama Dönem Proje Ödevi**

**MNIST Veri Seti ile Görüntü Sınıflandırma ve Tahmin Projesi**

**Hazırlayan: Taha Yunus Demir**

**Numara: 202213171811**

İçindekiler Tablosu

[1. Giriş 3](#_Toc183896215)

[2. Veri Seti 3](#_Toc183896216)

[2.1. Veri Seti Formatı 3](#_Toc183896217)

[3. Kullanılan Yöntem ve Araçlar 3](#_Toc183896218)

[3.1. Kullanılan Araçlar 3](#_Toc183896219)

[3.2. Yöntem 3](#_Toc183896220)

[4. Veri Ön İşleme 4](#_Toc183896221)

[4.1. Veri Dönüşümü 4](#_Toc183896222)

[4.2. Veri Yükleme 4](#_Toc183896223)

[5. Model Tasarımı 4](#_Toc183896224)

[5.1. Model Mimarisi 4](#_Toc183896225)

[5.2. Model Tanımlama 5](#_Toc183896226)

[6. Eğitim Süreci 6](#_Toc183896227)

[6.1. Hiperparametreler 6](#_Toc183896228)

[6.2. Eğitim Aşaması 6](#_Toc183896229)

[7. Değerlendirme ve Analiz 7](#_Toc183896230)

[7.1. Eğitim Kayıpları 7](#_Toc183896231)

[7.2. Performans Analizi 7](#_Toc183896232)

[7.3. Test Seti Üzerinde Tahminler 8](#_Toc183896233)

[8. Sonuçlar ve Çıkarımlar 8](#_Toc183896234)

[9. Sonuç Dosyası 8](#_Toc183896235)

[10. Kapanış 9](#_Toc183896236)

# 1. Giriş

Bu rapor, MNIST veri seti kullanılarak gerçekleştirilmiş bir görüntü sınıflandırma projesini kapsamaktadır. Proje kapsamında, el yazısı rakamları doğru bir şekilde sınıflandırmak için bir Convolutional Neural Network (CNN) modeli geliştirilmiştir. Modelin performansı değerlendirilmiş ve sonuçlar detaylı bir şekilde analiz edilmiştir.

Raporda, veri ön işleme, model tasarımı, eğitim süreci, sonuçların değerlendirilmesi ve elde edilen çıktıların ayrıntıları açıklanmaktadır. Ayrıca projede kullanılan yöntemlerin temel ilkeleri ve yazılım araçları hakkında bilgi verilmiştir.

# 2. Veri Seti

MNIST veri seti, el yazısı rakamların sınıflandırılması için yaygın olarak kullanılan bir benchmark veri setidir. Bu veri seti, 28x28 piksel boyutunda siyah-beyaz görüntülerden oluşmaktadır. Her bir görüntü, 0 ile 9 arasında bir rakamı temsil eder.

## 2.1. Veri Seti Formatı

* **Eğitim Veri Seti (train.csv)**: 42.000 örnek içerir. İlk sütun "label" olup, görüntülerin doğru etiketlerini içerir. Geri kalan 784 sütun, her bir pikselin yoğunluk değerini (0-255 arasında) içerir.
* **Test Veri Seti (test.csv)**: 28.000 örnek içerir. Bu veri setinde etiketler bulunmamaktadır ve tahmin yapılması beklenmektedir.

# 3. Kullanılan Yöntem ve Araçlar

## 3.1. Kullanılan Araçlar

* **PyTorch**: Model oluşturma, eğitim ve değerlendirme için kullanılmıştır.
* **Pandas**: Veri setlerini işlemek için kullanılmıştır.
* **Matplotlib & Seaborn**: Görselleştirme için tercih edilmiştir.

## 3.2. Yöntem

Proje kapsamında, bir Convolutional Neural Network (CNN) modeli geliştirilmiştir. CNN, görüntü işleme problemleri için özellikle etkili bir yöntemdir. Model tasarımı sırasında, aşağıdaki prensipler göz önünde bulundurulmuştur:

1. **Katmanlar**: Modelde üç adet konvolüsyonel katman ve iki tam bağlantılı katman bulunmaktadır.
2. **Düzenleme**: Regularization için Dropout kullanılmıştır.
3. **Aktivasyon Fonksiyonu**: ReLU aktivasyon fonksiyonu seçilmiştir.
4. **Normalizasyon**: Batch Normalization, ağın daha hızlı öğrenmesini ve daha kararlı hale gelmesini sağlamıştır.

# 4. Veri Ön İşleme

## 4.1. Veri Dönüşümü

Her bir görüntü, 784 sütundan (28x28 piksel) oluşan bir matris olarak yeniden şekillendirilmiştir. Görüntü verileri, normalize edilerek ağ girişine uygun hale getirilmiştir.

|  |
| --- |
| transform = transforms.Compose([  transforms.RandomRotation(20),  transforms.RandomAffine(0, shear=10, scale=(0.8, 1.2)),  transforms.ToTensor(),  transforms.Normalize((0.5,), (0.5,))  ]) |

## 4.2. Veri Yükleme

Özel bir Dataset sınıfı yazılarak veriler DataLoader kullanılarak modele uygun hale getirilmiştir.

|  |
| --- |
| # DataLoader  batch\_size = 64  train\_loader = DataLoader(train\_dataset, batch\_size=batch\_size, shuffle=True)  test\_loader = DataLoader(test\_dataset, batch\_size=batch\_size, shuffle=False)  print(f"Veri Seti Boyutları -> Eğitim: {len(train\_dataset)}, Test: {len(test\_dataset)}")  # Veri Seti Boyutları -> Eğitim: 42000, Test: 28000 |

# 5. Model Tasarımı

## 5.1. Model Mimarisi

Model, aşağıdaki katmanlardan oluşmaktadır:

* **Giriş Katmanı**: 28x28 boyutundaki görüntüleri alır.
* **Konvolüsyonel Katmanlar**: Görüntü özelliklerini öğrenir.
  + **Conv2d (32 filtre, 3x3 kernel)**
  + **Batch Normalization**
  + **MaxPooling**
* **Tam Bağlantılı Katmanlar**: Öğrenilen özelliklere göre sınıflandırma yapar.

**Model Mimarisi:**

|  |
| --- |
| Input -> Conv2d -> BatchNorm -> ReLU -> MaxPooling  -> Conv2d -> BatchNorm -> ReLU -> MaxPooling  -> Conv2d -> BatchNorm -> ReLU -> MaxPooling  -> Flatten -> Dense(256) -> Dropout -> Dense(10) |

## 5.2. Model Tanımlama

Aşağıdaki kod bloğu, modelin PyTorch kullanılarak tanımlanmasını göstermektedir:

|  |
| --- |
| class AdvancedCNN(nn.Module):  def \_\_init\_\_(self):  super(AdvancedCNN, self).\_\_init\_\_()  self.conv1 = nn.Conv2d(1, 32, kernel\_size=3, padding=1)  self.bn1 = nn.BatchNorm2d(32)  self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, padding=1)  self.bn2 = nn.BatchNorm2d(64)  self.conv3 = nn.Conv2d(64, 128, kernel\_size=3, padding=1)  self.bn3 = nn.BatchNorm2d(128)  self.pool = nn.MaxPool2d(2, 2)  self.fc1 = nn.Linear(128 \* 3 \* 3, 256)  self.fc2 = nn.Linear(256, 10)  self.dropout = nn.Dropout(0.4) |

# 6. Eğitim Süreci

## 6.1. Hiperparametreler

* Öğrenme oranı: 0.001
* Optimizasyon algoritması: Adam
* Kayıp fonksiyonu: Cross-Entropy Loss
* Epoch sayısı: 20
* Batch boyutu: 64

## 6.2. Eğitim Aşaması

Model, 20 epoch boyunca eğitilmiş ve her epoch sonunda kayıp değeri kaydedilmiştir.

|  |
| --- |
| Epoch 1/20, Ortalama Kayıp: 0.2588  Epoch 2/20, Ortalama Kayıp: 0.1089  Epoch 3/20, Ortalama Kayıp: 0.0896  Epoch 4/20, Ortalama Kayıp: 0.0820  Epoch 5/20, Ortalama Kayıp: 0.0702  Epoch 6/20, Ortalama Kayıp: 0.0611  Epoch 7/20, Ortalama Kayıp: 0.0605  Epoch 8/20, Ortalama Kayıp: 0.0561  Epoch 9/20, Ortalama Kayıp: 0.0535  Epoch 10/20, Ortalama Kayıp: 0.0535  Epoch 11/20, Ortalama Kayıp: 0.0504  Epoch 12/20, Ortalama Kayıp: 0.0462  Epoch 13/20, Ortalama Kayıp: 0.0456  Epoch 14/20, Ortalama Kayıp: 0.0423  Epoch 15/20, Ortalama Kayıp: 0.0424  Epoch 16/20, Ortalama Kayıp: 0.0405  Epoch 17/20, Ortalama Kayıp: 0.0415  Epoch 18/20, Ortalama Kayıp: 0.0388  Epoch 19/20, Ortalama Kayıp: 0.0383  Epoch 20/20, Ortalama Kayıp: 0.0366 |

# 7. Değerlendirme ve Analiz

## 7.1. Eğitim Kayıpları

Aşağıda, her bir epoch sonunda kaydedilen kayıp değerlerinin grafiği verilmiştir:

**Eğitim Kayıpları Grafiği**

metin, çizgi, ekran görüntüsü, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 7.2. Performans Analizi

Modelin performansı, eğitim seti üzerinde bir karışıklık matrisi ve sınıflandırma raporu ile değerlendirilmiştir.

**Karışıklık Matrisi**

metin, ekran görüntüsü, diyagram, kare içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Sınıflandırma Raporu**

|  |
| --- |
| precision recall f1-score support  0 1.00 1.00 1.00 4132  1 0.99 1.00 1.00 4684  2 0.99 0.99 0.99 4177  3 1.00 0.98 0.99 4351  4 0.99 0.99 0.99 4072  5 0.99 0.98 0.99 3795  6 1.00 0.99 0.99 4137  7 0.98 0.99 0.99 4401  8 0.99 0.99 0.99 4063  9 0.98 0.99 0.98 4188  accuracy 0.99 42000  macro avg 0.99 0.99 0.99 42000  weighted avg 0.99 0.99 0.99 42000 |

## 7.3. Test Seti Üzerinde Tahminler

Test veri seti üzerinde yapılan tahminlerin sonuçları sonuç.csv dosyasına kaydedilmiştir.

# 8. Sonuçlar ve Çıkarımlar

* **Başarı Oranı**: Eğitim seti üzerinde yüksek doğruluk elde edilmiştir.
* **Görselleştirme**: Karışıklık matrisi ve kayıp grafikleri, modelin etkili bir şekilde öğrenim sağladığını göstermektedir.
* **İyileştirme Önerileri**: Veri artırma teknikleri ve daha derin modeller ile doğruluk artırılabilir.

# 9. Sonuç Dosyası

Test seti için yapılan tahminlerin sonuçları şu formatta oluşturulmuş ve sonuç.csv dosyasına kaydedilmiştir:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | ImageId,Label | | 1,2 | | 2 | | 3,9 | | 4  .  .  . | |

# 10. Kapanış

Bu proje, el yazısı rakam sınıflandırması için güçlü bir CNN modeli geliştirilmesi ve değerlendirilmesi sürecini kapsamaktadır. Tüm süreç açık bir şekilde belgelenmiş ve sonuçlar detaylı bir şekilde analiz edilmiştir. Kodlar ve rapor, GitHub’a yüklenmiştir.