T.C.
KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YÜKSEK DÜZEY PROGRAMLAMA **DERSİ** PROJE RAPORU

Digit Recognizer Veri Seti ile Model **Geliştirme**

İpek Melisa **Yılmaz**
202113172023

EĞİTMEN: Doç.Dr. Hasan Temurtaş

1. Proje **Amacı** ve Genel **Bakış**

Bu projede, MNIST veri kümesi kullanılarak bir derin öğrenme modeli oluşturulmuştur. Modelin amacı, el yazısı rakamları (0-9) doğru bir şekilde sınıflandırmaktır. Proje, görüntü verilerini işleyerek sınıflandırma yapmak için bir yapay sinir ağı (ANN) kullanmaktadır.

2. Veri Kümesi

Veri kümesi, MNIST veri setinden alınmıştır. Bu veri seti, el yazısı ile yazılmış 28x28 piksel boyutlarında 60.000 eğitim ve 10.000 test görüntüsünden oluşmaktadır. Her görüntü bir rakamı (0'dan 9'a kadar) temsil etmektedir.

X_train: Eğitim verisi (60.000 görüntü).
y_train: Eğitim etiketleri (60.000 etiket).
X_test: Test verisi (10.000 görüntü).
y_test: Test etiketleri (10.000 etiket).

3. Verilerin Yüklenmesi ve Ön **İşleme**

Verilerin Yüklenmesi
Proje ilk adımda, MNIST veri kümesi pandas ile CSV dosyalarından yüklenmiştir:

Veri Setinin Yüklenmesi

In [105…
```
mnist_dataset=pd.read_csv("/kaggle/input/digitrecognizer/train.csv")
mnist_dataset.shape
```

Out[105… (42000, 785)

Eğitim ve test veri kümeleri pandas DataFrame'lerine yüklenmiştir. Burada train.csv eğitim verisini ve test.csv test verisini içermektedir.

Verilerin Düzenlenmesi ve Normalizasyon

Eğitim verisi ve test verisi, modelin gereksinimlerine göre işlenmiştir:

```
X_train = mnist_dataset_train.drop(columns=["label"]).values
y_train = mnist_dataset_train["label"].values
X_test = mnist_dataset_test.values
```

Eğitim verisinden etiketler (label) ayrılmış ve görüntüler 28x28 boyutlarından bir vektöre dönüştürülmüştür. Ayrıca, görüntüler 0-255 aralığında değerler aldığından, bu değerler 0 ile 1 arasına normalize edilmiştir.

In [126…
```
X_train=X_train/255
X_test=X_test/255
```

4. Modelin **Oluşturulması**

Model, bir yapay sinir ağı (ANN) kullanarak oluşturulmuştur. Modelin yapısı aşağıdaki gibi belirlenmiştir:

Giriş Katmanı: 784 nöron (28x28 piksel görüntülerin düzleştirilmiş hali).
Gizli Katman: 128 nöron ve ReLU aktivasyon fonksiyonu.
Çıktı Katmanı: 10 nöron (0'dan 9'a kadar olan rakamlar) ve Softmax aktivasyon fonksiyonu.

İlk katman: 128 nöronlu bir Dense katmanı eklenmiştir ve ReLU aktivasyon fonksiyonu kullanılmıştır. Bu katman, doğrusal olmayan özellikleri öğrenmeye yardımcı olur.

Çıktı katmanı: 10 nöronlu bir Dense katmanı eklenmiştir ve Softmax aktivasyon fonksiyonu kullanılmıştır. MNIST veri setindeki 10 sınıfı temsil eden bu katman, her sınıf için bir olasılık değeri döndürür.

```
In [129…  model=Sequential()

In [130…  model.add(Dense(128,activation='relu'))

In [131…  model.add(Dense(10,activation='softmax'))

In [132…  model.summary()
```

Model Özeti

Modelin özetine bakıldığında, ilk katmanın 128 nörona sahip olduğu ve çıkış katmanının 10 nörona sahip olduğu görülür. Çıkış katmanı, Softmax fonksiyonu ile her bir sınıf için olasılıkları hesaplar.

5. Modelin Derlenmesi

Model derlenmeden önce, modelin eğitilebilmesi için compile fonksiyonu çağrılmalıdır. Bu fonksiyon, kayıp fonksiyonu, optimizasyon algoritması ve değerlendirme metriğini belirtir.

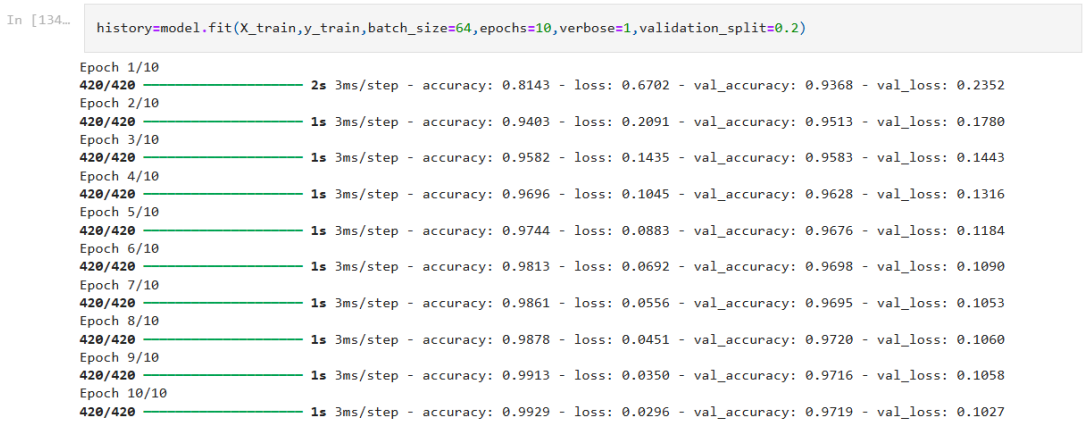
```
In [133…  model.compile(optimizer='Adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])

In [134…  history=model.fit(X_train,y_train,batch_size=64,epochs=10,verbose=1,validation_split=0.2)

Epoch 1/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8143 - loss: 0.6702 - val_accuracy: 0.9368 - val_loss: 0.2352
Epoch 2/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9403 - loss: 0.2091 - val_accuracy: 0.9513 - val_loss: 0.1780
Epoch 3/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9582 - loss: 0.1435 - val_accuracy: 0.9583 - val_loss: 0.1443
Epoch 4/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9696 - loss: 0.1045 - val_accuracy: 0.9628 - val_loss: 0.1316
Epoch 5/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9744 - loss: 0.0883 - val_accuracy: 0.9676 - val_loss: 0.1184
Epoch 6/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9813 - loss: 0.0692 - val_accuracy: 0.9698 - val_loss: 0.1090
Epoch 7/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9861 - loss: 0.0556 - val_accuracy: 0.9695 - val_loss: 0.1053
Epoch 8/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9878 - loss: 0.0451 - val_accuracy: 0.9720 - val_loss: 0.1060
Epoch 9/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9913 - loss: 0.0350 - val_accuracy: 0.9716 - val_loss: 0.1058
Epoch 10/10
420/420 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9929 - loss: 0.0296 - val_accuracy: 0.9719 - val_loss: 0.1027
```

Optimizer: Adam (önerilen bir optimizasyon algoritması).
Loss Function: Kategorik çapraz entropi kaybı (çok sınıflı sınıflandırma problemi için).

Metric: Modelin başarısını değerlendirmek için doğruluk (accuracy).

6. Modelin **Eğitilmesi**

Modelin eğitilmesi için fit fonksiyonu kullanılmıştır. Bu fonksiyon, modelin eğitim verisi üzerinde eğitilmesini sağlar. Eğitimde, her bir batch için 64 örnek kullanılacak ve toplamda 10 epoch boyunca eğitim yapılacaktır.



batch_size: 64 (her adımda 64 örnek işlenecektir).

epochs: 10 (model 10 kez tüm eğitim verisi üzerinde çalıştırılacaktır).

validation_split: Eğitim verilerinin %20'si doğrulama için ayrılacaktır.

7. Modelin **Performansının Değerlendirilmesi**

Modelin doğruluğu, test verisi üzerinde değerlendirilmiştir:

```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose

print(f"Test Accuracy: {test_acc}")
```

Bu işlem, modelin test verisi üzerindeki doğruluğunu hesaplar. Sonuç olarak, modelin genel başarısı hakkında bilgi edinilir.

8. Tahminlerin **Yapılması**

Eğitim tamamlandıktan sonra, test veri kümesi üzerinde tahminler yapılmıştır:

In [145…

```
preds = model.predict(mnist_dataset_test)
results = pd.DataFrame(preds)
results['label'] = results.idxmax(axis = 1)
results['row_index'] = results.index +1
df = pd.DataFrame({
    'ImageId': results['row_index'],
    'label': results['label']
})
```

```
875/875 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
```

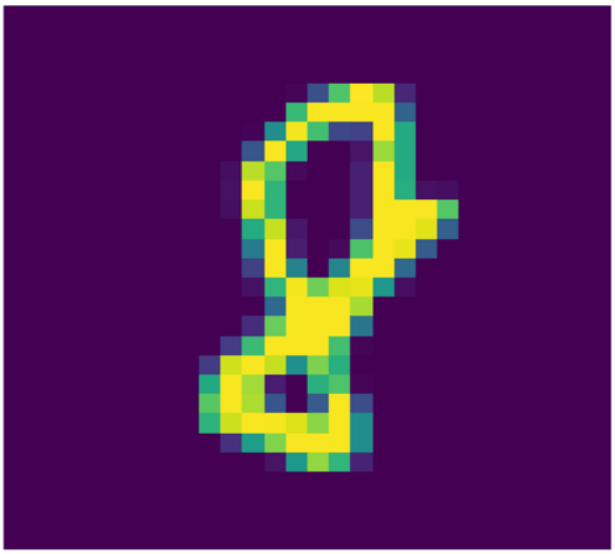
Burada, modelin tahmin ettiği olasılıkları içeren bir DataFrame oluşturulmuş ve her örnek için en yüksek olasılığa sahip sınıf etiketi seçilmiştir.

9. Sonuçlar ve **Çıktılar**

Model, test veri kümesi üzerinde yapılan tahminlerde yüksek doğruluk oranlarına ulaşmıştır. df DataFrame'i, her bir görüntü için tahmin edilen etiketleri içerir. Bu sonuçlar Kaggle gibi platformlarda test verisi üzerinde değerlendirilebilir.

Test Sonuçları:
Modelin doğruluğu test seti üzerinde belirli bir seviyede çıkmıştır ve her bir test görüntüsü için tahmin edilen etiketler şu şekilde sıralanabilir.

*Görüntü **Örneği:***



```
      label
5457      8
<class 'numpy.ndarray'>
```

Bu örnekte model, görüntüyü doğru bir şekilde 8 olarak sınıflandırmıştır.

10. Sonuç

Bu proje, derin öğrenme tekniklerini kullanarak, MNIST veri setindeki el yazısı rakamlarını başarılı bir şekilde sınıflandırmayı başarmıştır. Model, test veri kümesi üzerinde yüksek doğruluk oranları elde etmiştir. Gelecekte, modelin doğruluğunu artırmak için farklı yapılandırmalar veya hiperparametre ayarlamaları yapılabilir.