

## YÜKSEK DÜZEY PROGRAMLAMA PROJE ÖDEVİ

AD : Tuğçe

SOYAD : Özturaç

OKUL NUMARASI : 202113171076

seçtiğim veri seti--**Digit Recognizer**: <https://www.kaggle.com/competitions/digit-recognizer/data>

### Proje Girişi:

Digit Recognizer, el yazısı ile yazılmış rakamların bulunduğu bir veri setidir. Bu proje kapsamında, katılımcılar aşağıdaki adımları takip ederek bir model geliştireceklerdir:

Veri Setinin Seçimi ve Yüklenmesi: Digit Recognizer veri seti, her bir görüntüde 28x28 piksel boyutunda el yazısı rakamlarını içerir. Bu veri seti, Kaggle üzerinden temin edilebilir. Eğitim veri seti, 60,000 örnek içerirken, test veri seti 10,000 örnekten oluşmaktadır.

### Veri Ön İşleme:

Görüntülerin normalizasyonu yapılacak (0-255 arası değerlerin 0-1 arasına çekilmesi).

Reshape işlemi ile görüntüler 28x28 boyutlarından uygun formatta modele verilecek hale getirilir.

Eksik veya hatalı veriler kontrol edilir ve işlenir.

### Model Eğitimi:

Derin öğrenme tabanlı Convolutional Neural Network (CNN) modeli kullanılacaktır. CNN, görüntü işleme ve sınıflandırma için yaygın olarak kullanılan bir modeldir.

Modelde birden fazla evrişim katmanı (Conv2D), max pooling katmanı, ve sonrasında tam bağlantılı katmanlar (Dense) yer alacaktır.

Modelin Test Edilmesi:

Test veri seti üzerinde modelin başarımlı doğruluk (accuracy) ölçülerek değerlendirilir.

Eğitilen modelin kaybı (loss) ve doğruluğu eğitim sürecinde izlenir ve görselleştirilir.

Sonuçların Görselleştirilmesi:

Modelin doğruluğunu ve kaybını gösteren grafikler çizilecektir.

Test veri setinden örnekler seçilip, modelin tahminleri görsel olarak karşılaştırılacaktır.

Projenin amacı, el yazısı rakamları sınıflandırmak için bir makine öğrenimi modeli (özellikle CNN) geliştirmek ve bu modelin doğruluğunu test etmektir. Bu süreç, görüntü işleme ve sınıflandırma algoritmalarını kullanarak, modelin el yazısı rakamlarını doğru bir şekilde tanıyabilmesini sağlamak üzerine odaklanacaktır.

## PROJE KODLARI VE AÇIKLAMALAR:

gerekli kütüphaneleri içe aktaran kod parçası

```
[100]: # Gerekli kütüphaneler
import pandas as pd      # Veri işleme
import numpy as np       # Matematiksel işlemler
import matplotlib.pyplot as plt # Görselleştirme
import seaborn as sns    # İleri görselleştirme
```

## üç farklı CSV dosyasını pandas ile yükleyen kod parçası

```
[98]: # CSV dosyalarını yükleme
train_df = pd.read_csv("train.csv") # Eğitim veri seti
test_df = pd.read_csv("test.csv") # Test veri seti
submission_df = pd.read_csv("sample_submission.csv") # Örnek çıktı dosyası
```

## veri setini incelemek için birkaç işlem yapan kod parçası

```
[7]: # İlk birkaç satırı görüntüleme
print(train_df.head())

# Veri seti hakkında bilgi
print(train_df.info())

# Eksik değerlerin kontrolü
print(train_df.isnull().sum())
|
```

```
print(train_df.isnull().sum())

# Eksik değerlerin kontrolü
print(train_df.isnull().sum())
```

```
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
```

```
[5 rows x 785 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42000 entries, 0 to 41999
Columns: 785 entries, label to pixel783
dtypes: int64(785)
memory usage: 251.5 MB
None
label 0
pixel0 0
pixel1 0
pixel2 0
pixel3 0
..
pixel779 0
pixel780 0
pixel781 0
pixel782 0
pixel783 0
Length: 785, dtype: int64
```

```
[180]: # Veri setindeki eksik değerleri kontrol et
print(train_df.isnull().sum()) # Eğitim veri setinde eksik değerler
print(test_df.isnull().sum()) # Test veri setinde eksik değerler

label      0
pixel0     0
pixel1     0
pixel2     0
pixel3     0
..
pixel1779  0
pixel1780  0
pixel1781  0
pixel1782  0
pixel1783  0
Length: 785, dtype: int64
pixel0     0
pixel1     0
pixel2     0
pixel3     0
pixel4     0
..
pixel1779  0
pixel1780  0
pixel1781  0
pixel1782  0
pixel1783  0
Length: 784, dtype: int64
```

```
[15]: # Tabloyu daha okunabilir bir şekilde göstermek için stil ekleme
train_df.head(10).style.set_table_styles(
    [{"selector": "th", "props": [{"background-color", "#f2f2f2"}, {"font-weight", "bold"}]},
     {"selector": "td", "props": [{"border", "1px solid #ccc"}]})
```

eğitim veri setindeki sütunların isimlerini liste halinde gösterir:

TensorFlow kütüphanesini Python ortamınıza yükler.

```

[87]: !pip install tensorflow

Requirement already satisfied: tensorflow in c:\users\routu\anaconda3\lib\site-packages (2.18.0)
Requirement already satisfied: tensorflow-intel==2.18.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow) (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (24.3.25)
Requirement already satisfied: gast<=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (24.1)
Requirement already satisfied: protobuf<4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (4.25.3)
Requirement already satisfied: requests>=2.21.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.68.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in c:\users\routu\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.7.0)

```

eğitim ve test veri setlerini hazırlamak için

TensorFlow ve Keras kullanarak MNIST veri setini yükler ve ön işler.

```
==2.18.0->tensorflow) (0.1.0)

[146]: # Eğitim verisini ayırma
X_train = train_df.drop('label', axis=1).values # 'label' dışındaki sütunlar
y_train = train_df['label'].values # 'label' hedef sütunu

# Eğer test etiketleri yoksa:
# Test verisi yalnızca özellikleri içerir
X_test = test_df.values # Eğer 'label' sütunu yoksa doğrudan test_df kullanılır

# Reshape işlemi: Görüntü formatı (28x28x1)
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1) # Eğitim verisi
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1) # Test verisi

# Normalizasyon (0-255 değerlerini 0-1 arasına çekme)
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255

[3]: import tensorflow as tf
from tensorflow.keras import layers, models
import pandas as pd
import numpy as np

# TensorFlow Keras MNIST veri setini yükleyelim (Eğer kendi veri setinizi kullanıyorsanız, burada yükleme işlemi farklı olur)
(train_data, train_labels), (test_data, test_labels) = tf.keras.datasets.mnist.load_data()

# Veriyi normalize etme (0-255 arası değerleri 0-1 arası değerler yapmak)
train_data = train_data / 255.0
test_data = test_data / 255.0

# Veriyi şekillendirme (28x28'lik görüntüler)
train_data = train_data.reshape((train_data.shape[0], 28, 28, 1)) # (60000, 28, 28, 1)
test_data = test_data.reshape((test_data.shape[0], 28, 28, 1)) # (10000, 28, 28, 1)
```

bir Konvolüsyonel Sinir Ağı (CNN) modeli oluşturur ve derler.

```
[136]: # CNN Modeli oluşturma
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)), # İlk evrişim katmanı
    layers.MaxPooling2D((2, 2)), # MaxPooling ile boyut küçültme
    layers.Conv2D(64, (3, 3), activation='relu'), # İkinci evrişim katmanı
    layers.MaxPooling2D((2, 2)), # MaxPooling
    layers.Conv2D(64, (3, 3), activation='relu'), # Üçüncü evrişim katmanı
    layers.Flatten(), # Düzleştirme katmanı (full connected katmanlar için)
    layers.Dense(64, activation='relu'), # Tam bağlantılı katman
    layers.Dense(10, activation='softmax') # 10 sınıf (rakamlar 0-9)
])

# Modeli derleme (optimizer ve loss fonksiyonu seçiyoruz)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Modelin özetini yazdırma
model.summary()
```

C:\Users\roztu\anaconda3\Lib\site-packages\Keras\src\layers\convolutional\base\_conv.py:107: UserWarning: Do not pass an `input\_shape` / `input\_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.  
super().\_\_init\_\_(activity\_regularizer=activity\_regularizer, \*\*kwargs)

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_4 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_5 (Conv2D)	(None, 3, 3, 64)	36,928
flatten_1 (Flatten)	(None, 576)	0
dense_2 (Dense)	(None, 64)	36,928
dense_3 (Dense)	(None, 10)	650

Total params: 93,322 (364.54 KB)  
Trainable params: 93,322 (364.54 KB)  
Non-trainable params: 0 (0.00 B)

modelin eğitim sürecini başlatır ve belirtilen sayıda epoch (10 epoch) boyunca verileri kullanarak eğitir:

eğitilen modelin test verisi üzerinde nasıl performans gösterdiğini değerlendirir:

```
[138]: # Modeli eğitime (10 epoch boyunca)
model.fit(train_data, train_labels, epochs=10, batch_size=64)

Epoch 1/10
938/938 — 3s 3ms/step - accuracy: 0.8743 - loss: 0.4123
Epoch 2/10
938/938 — 2s 3ms/step - accuracy: 0.9839 - loss: 0.0530
Epoch 3/10
938/938 — 3s 3ms/step - accuracy: 0.9889 - loss: 0.0340
Epoch 4/10
938/938 — 3s 3ms/step - accuracy: 0.9917 - loss: 0.0255
Epoch 5/10
938/938 — 3s 3ms/step - accuracy: 0.9928 - loss: 0.0213
Epoch 6/10
938/938 — 3s 3ms/step - accuracy: 0.9948 - loss: 0.0168
Epoch 7/10
938/938 — 3s 3ms/step - accuracy: 0.9955 - loss: 0.0133
Epoch 8/10
938/938 — 3s 3ms/step - accuracy: 0.9966 - loss: 0.0109
Epoch 9/10
938/938 — 2s 3ms/step - accuracy: 0.9961 - loss: 0.0118
Epoch 10/10
938/938 — 3s 3ms/step - accuracy: 0.9966 - loss: 0.0098

[138]: <keras.src.callbacks.history.History at 0x259ad1fa540>

[110]: # Modeli test verisiyle değerlendirme
test_loss, test_acc = model.evaluate(test_data, test_labels)

print(f'Test accuracy: {test_acc}')

313/313 — 0s 1ms/step - accuracy: 0.9847 - loss: 0.0446
Test accuracy: 0.9884999990463257
```

modelin test verisi üzerinde yaptığı tahminleri inceleyip görselleştirir:

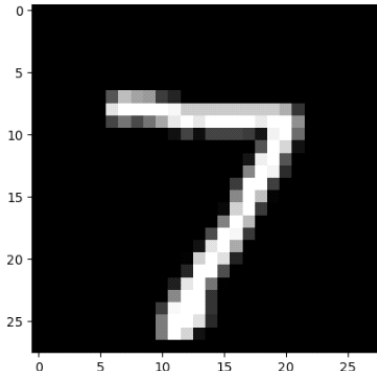
```
[112]: # Test verisinden bir örnek seçip tahmin yapalım
predictions = model.predict(test_data)

# İlk test örneğini ve modelin tahminini gösterelim
print(f'Tahmin: {np.argmax(predictions[0])}')
print(f'Gerçek Değer: {test_labels[0]}')

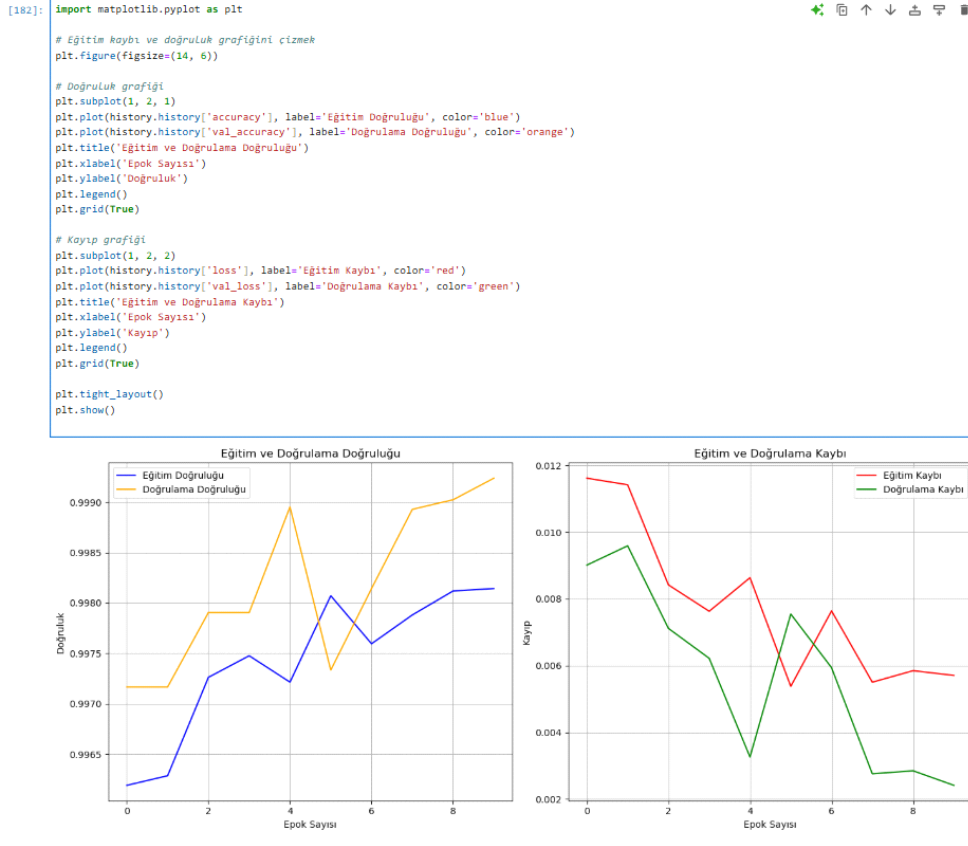
# İlk test örneğini görselleştirelim
import matplotlib.pyplot as plt

plt.imshow(test_data[0].reshape(28, 28), cmap='gray')
plt.show()

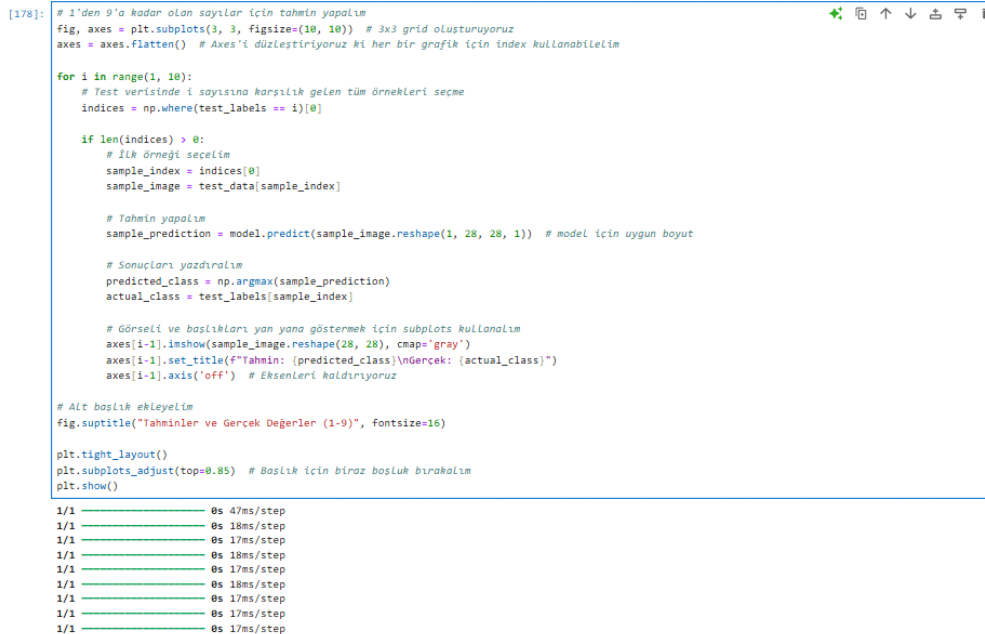
313/313 — 0s 1ms/step
Tahmin: 7
Gerçek Değer: 7
```



eğitim sürecinin kaybı (loss) ve doğruluğu (accuracy) hakkında grafikler çizer:

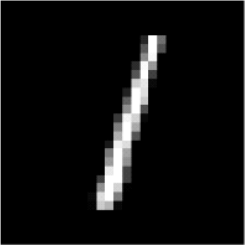

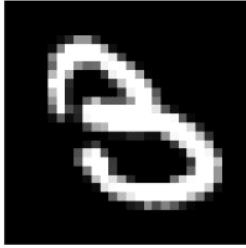
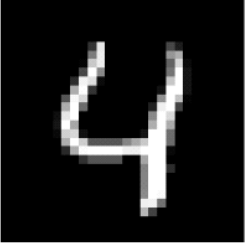


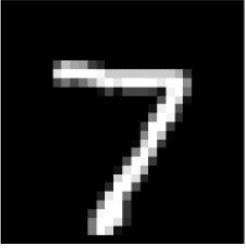

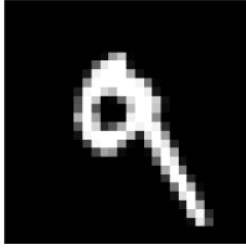


1'den 9'a kadar olan rakamlar için test veri setinden örnekler seçer ve modelin tahminlerini gerçek değerlerle karşılaştırarak görselleştirir:





## Tahminler ve Gerçek Değerler (1-9)

Tahmin: 1 Gerçek: 1	Tahmin: 2 Gerçek: 2	Tahmin: 3 Gerçek: 3
		
Tahmin: 4 Gerçek: 4	Tahmin: 5 Gerçek: 5	Tahmin: 6 Gerçek: 6
		
Tahmin: 7 Gerçek: 7	Tahmin: 8 Gerçek: 8	Tahmin: 9 Gerçek: 9
		

Projenin sonucu, seçilen veri seti üzerinde eğitim ve test edilen bir modelin doğruluk oranının belirlenmesidir. Model, verilen veriye göre doğru sınıflandırmalar yaparak yüksek bir başarıya ulaşmayı hedefler. Sonuç olarak, modelin performansı değerlendirilir ve test verisi üzerindeki doğruluk oranı raporlanır.