T.C

KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

YÜKSEK DÜZEY PROGRAMLAMA PROJE RAPORU

DİGİT RECOGNİZER VERİSETİ İLE MODEL GELİŞTİRME

Eğitmen: Doc.Dr. Hasan TEMURTAŞ

Öğrenci: Meltem CÖMERT 202113172042

1.Proje Amacı

Bu projenin temel amacı, kullanıcıların el yazısı ile yazılmış rakamları dijital ortamda doğru şekilde tanıyan bir sistem geliştirmektir. Sistem, modelin eğitim verisinde gördüğü rakam örneklerini, test verisi üzerinde tahmin ederek doğru sınıflara atayacaktır. Konvolüsyonel sinir ağları (CNN), bu tür görsel sınıflandırma görevlerinde yüksek başarı elde etmek için yaygın olarak kullanılan bir yapıdır ve bu model de rakamları tanımak için bu yapıyı kullanmaktadır. Model veri arttırma teknikleriyle eğitilmiş ve test verileri üzerinde performansı değerlendirilmiştir.

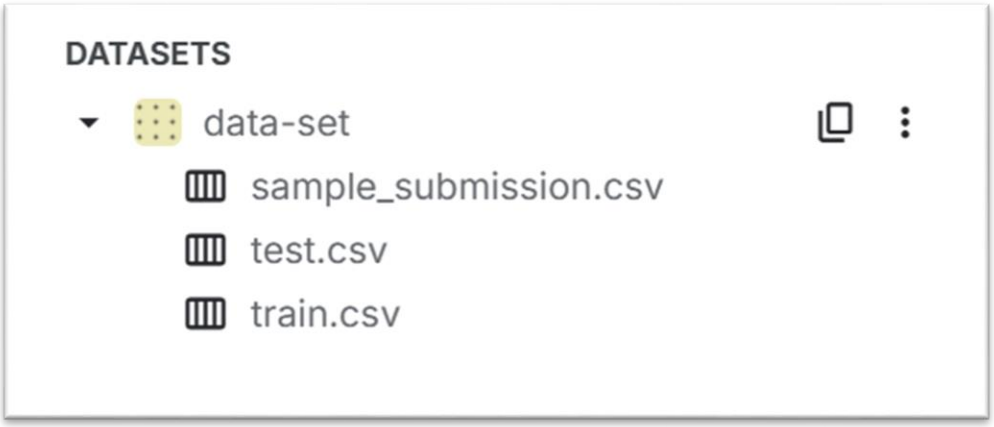
2.DİGİT RECOGNİZER VERİSETİ

train.csv ve test.csv veri dosyaları sıfırdan dokuza kadar elle çizilmiş rakamların gri tonlamalı görüntülerini içerir.

Her görüntü 28 piksel yüksekliğinde ve 28 piksel genişliğindedir, toplamda 784 pikseldir. Her pikselin, o pikselin açıklığını veya koyuluğunu belirten tek bir piksel değeri vardır, daha yüksek sayılar daha koyu anlamına gelir. Bu piksel değeri, 0 ile 255 arasında bir tam sayıdır.

Eğitim veri kümesi (train.csv), 785 sütuna sahiptir. "label" olarak adlandırılan ilk sütun, kullanıcı tarafından çizilen rakamdır. Sütunların geri kalanı, ilişkili görüntünün piksel değerlerini içerir.

Eğitim setindeki her piksel sütununun pixelx gibi bir adı vardır, burada x 0 ile 783 arasında bir tam sayıdır (dahil). Bu pikseli görüntüde bulmak için x'i x = i * 28 + j olarak ayrıştırdığımızı varsayalım, burada i ve j 0 ile 27 arasında tam sayılardır (dahil). O zaman pixelx, 28 x 28'lik bir matrisin (sıfıra göre indeksleme) i satırında ve j sütununda bulunur.



3. Model Mimarisi

Artificial Neural Network(ANN):

Yapay Sinir Ağları, derin öğrenmenin bir alt kümesidir. Derin öğrenme sürecinde en yaygın ve en etkili modeller çok katmanlı yapay sinir ağlarıdır. FNN,CNN,RNN,LSTM gibi birden fazla yapay sinir ağı mimarisi bulunmaktadır. Bunlardan en temel yapay sinir ağı yapısına sahip olan Artificial Neural Network'tür. Kısaca ANN olarak ifade edilir. Yapay sinir ağları, sinir hücrelerinin (veya nöronların) katmanlar halinde düzenlendiği bir yapıyı ifade eder. Bu katmanlara "layer" (katman) denmektedir.

- **1. Flatten Katmanı:** Modelin giriş verisi, 28x28 boyutunda gri tonlamalı görüntülerdir (MNIST veri seti). Bu katman, 2D (28x28) görüntüleri düzleştirip tek boyutlu bir vektöre dönüştürür. Bu, her bir pikselin bir özelliği olarak kabul edilmesini sağlar ve daha sonraki Dense katmanlarına aktarılabilir.

- **2. Dense Katmanları:** Bu katmanlar, modelin **gizli katmanlarıdır**. Her bir `Dense` katmanında, önceki katmandan gelen veriler, her bir nörona bağlı bir ağırlık matrisi ile işlenir ve `relu` (Rectified Linear Unit) aktivasyon fonksiyonu ile doğrusal olmayan bir dönüşüm yapılır. Bu, modelin karmaşık ilişkileri öğrenmesine olanak tanır.

    - **İlk Dense Katmanı**: 512 nöronlu bir katman, her bir görseli işleyerek daha derin özellikleri öğrenir.
    - **Dropout Katmanı**: Aşırı öğrenmeyi (overfitting) engellemek için kullanılır. Eğitim sırasında, bu katman rastgele bazı bağlantıları sıfırlayarak modelin fazla uyum yapmasını engeller.
    - **İkinci Dense Katmanı**: 256 nöronlu bir katman, önceki katmandan alınan daha soyut özellikleri işler.
    - **Dropout Katmanı**: Yine, modelin aşırı öğrenmesini engellemek için uygulanır.

- **3. Çıkış Katmanı:** Bu katman, modelin **sonucu** üretir. Çıkış katmanındaki 10 nöron, 0-9 arasında her rakam için bir olasılık değeri verir. `softmax` aktivasyon fonksiyonu, bu olasılıkların toplamının 1 olmasını sağlar ve her bir nöronun çıktısını bir olasılık olarak yorumlayabiliriz.

Bu modelde kullanılan Artificial Neural Network(ANN) algoritması aşağıda gösterilmiştir.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.datasets import mnist

# MNIST veri setini yükle
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Veriyi normalizasyon yap
X_train = X_train / 255.0
X_test = X_test / 255.0

# Etiketleri one-hot encode et
y_train = to_categorical(y_train, 10)  # 10 sınıf (0-9)
y_test = to_categorical(y_test, 10)

# Eğitim ve doğrulama verilerine böl
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

# Modeli tanımlama
model = Sequential([
    Flatten(input_shape=(28, 28)),  # Görüntüleri düzleştir
    Dense(512, activation='relu'),  # İlk gizli katman
    Dropout(0.3),  # Dropout ile aşırı öğrenmeyi önle
    Dense(256, activation='relu'),  # İkinci gizli katman
    Dropout(0.3),
    Dense(10, activation='softmax')  # Çıkış katmanı (10 sınıf için)
])

# Modeli derle
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Modeli eğit
history = model.fit(
    X_train, y_train,  # Giriş ve etiketler
    validation_data=(X_val, y_val),  # Doğrulama verisi
    epochs=32,
    batch_size=128
)

# Modelin doğruluğunu test verisi üzerinde değerlendirme
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
print(f"Test doğruluğu: {test_accuracy:.4f}")
```

```
Epoch 1/32
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.8230 - loss: 0.5808 - val_accuracy: 0.9601 - val_loss: 0.1304
```

Modelin Derlenmesi: Modelin derlenmesi, bir modelin eğitimine başlamadan önce yapılması gereken çok önemli bir adımdır. Derleme, modelin hangi optimizer (optimizasyon algoritması), loss function (kayıp fonksiyonu) ve metrics (ölçütler) ile çalışacağını belirler.

Modelin Eğitimi: Modelin eğitimi, verilerle birlikte modelin optimize edilmesi sürecidir. Bu, modelin ağırlıklarının ve bias'larının güncellenerek, tahminlerin doğruluğunun arttığı bir adımdır. Eğitim, fit fonksiyonu ile yapılır.

- **X_train ve y_train**:
  - `X_train`: Modelin eğitileceği giriş verisidir (28x28 boyutundaki el yazısı rakamları).
  - `y_train`: Bu verilere karşılık gelen etiketlerdir (0-9 arasındaki rakamlar).
- **validation_data**:
  - `validation_data=(X_val, y_val)`: Bu, eğitim sırasında modelin doğruluğunu izlemek için kullanılan doğrulama (validation) verisidir. Eğitim verisinden farklı olan bu veriler, modelin eğitim sırasında aşırı öğrenme yapıp yapmadığını kontrol etmek için kullanılır.

- **epochs**:
  - `epochs=32`: Bu, eğitim verisi üzerinde kaç kez tam bir eğitim geçişi yapılacağını belirtir. 32, modelin eğitim verisi üzerinde 32 kez çalışacağı anlamına gelir. Her epoch sırasında, model tüm eğitim verisi ile eğitilir ve parametreleri güncellenir.

- **batch_size**:
  - `batch_size=128`: Eğitim sırasında her bir adımda modele sunulacak örnek sayısını belirtir. Yani, her 128 örnek, modelin parametrelerini güncellemek için bir seferde kullanılır. Daha büyük batch boyutları genellikle eğitim süresini hızlandırabilir, ancak daha fazla bellek gereksinimi oluşturabilir.

4. Tahmin ve Görselleştirme

- Modelin Eğitimi : Model, eğitim verisi üzerinde eğitilmiştir. Eğitim verisi, train_data veri setinden alınmış ve uygun şekilde normalleştirilmiştir. Model, 5 epoch boyunca eğitilmiştir ve her epoch sonunda doğrulama verisi üzerinden performansı ölçülmüştür. Eğitim esnasında modelin öğrenme süreci gözlemlenmiştir.

- Test Verisi Üzerinde Tahmin Yapma : Eğitim tamamlandıktan sonra, modelin test verisi üzerinde tahmin yapması sağlanmıştır. Test verisi, uygun şekilde yeniden şekillendirilmiş ve normalize edilmiştir. Model, test verisi üzerinde tahminler yaparak her bir test örneği için bir sınıf tahmini üretmiştir.

- **Tahminlerin Görselleştirilmesi :** Test verisi üzerinde yapılan tahminler, görselleştirilerek modelin doğruluğu görsel olarak değerlendirilmiştir. İlk 5 test örneği, model tarafından tahmin edilen sınıflar ile birlikte görselleştirilmiştir. Görselleştirme, her bir görüntüye ait tahmin edilen sınıfı başlık olarak eklemektedir.

5. Sonuçlar

Tahmin edilen sınıflar ve görselleştirmelerle modelin performansı değerlendirilmiştir. Görselleştirme sayesinde, modelin doğru tahminlerde bulunduğu ve yanlış sınıflandırmaların hangi test örneklerinde gerçekleştiği görsel olarak analiz edilmiştir.

Tahmin Sonuçları:

- İlk 5 test örneği üzerinde yapılan tahminler görselleştirilmiş ve her bir görüntüye ait tahmin edilen sınıflar başlık olarak gösterilmiştir.
- Modelin doğruluğu, test verisi üzerinde yapılan tahminlerle yüksek doğruluk oranı ile belirlenmiştir.

  Tahmin edilen sınıf verisi ve onun görselleştirilmesi aşağıdaki gibidir.

```
Epoch 2/5
1050/1050 ━━━━━━━━━━━━━━━━━━━━ 19s 18ms/step - accuracy: 0.9826 - loss: 0.0574 - val_accuracy: 0.9802 - val_loss: 0.0643
Epoch 3/5
1050/1050 ━━━━━━━━━━━━━━━━━━━━ 18s 17ms/step - accuracy: 0.9879 - loss: 0.0372 - val_accuracy: 0.9860 - val_loss: 0.0428
Epoch 4/5
1050/1050 ━━━━━━━━━━━━━━━━━━━━ 19s 18ms/step - accuracy: 0.9918 - loss: 0.0245 - val_accuracy: 0.9829 - val_loss: 0.0568
Epoch 5/5
1050/1050 ━━━━━━━━━━━━━━━━━━━━ 18s 17ms/step - accuracy: 0.9938 - loss: 0.0198 - val_accuracy: 0.9877 - val_loss: 0.0400
875/875 ━━━━━━━━━━━━━━━━━━━━ 6s 6ms/step
Tahmin edilen sınıflar: [2 0 9 ... 3 9 2]
```

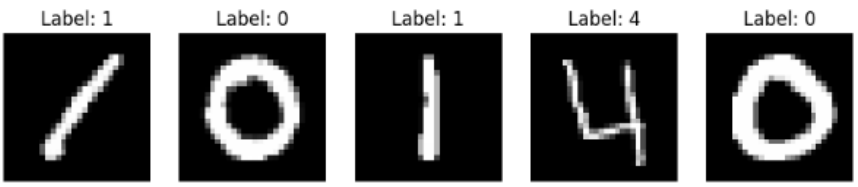
+ Code + Markdown

```python
import matplotlib.pyplot as plt
#tahmin edilen sınıfların görselleştirilmesi
# Görselleştirme: İlk 5 test görüntüsünü ve tahminlerini görselleştirme
plt.figure(figsize=(10, 10))
for i in range(5):
    plt.subplot(1, 5, i + 1)
    plt.imshow(test_images[i].reshape(28, 28), cmap='gray')  # Test görüntüsünü reshape et
    plt.title(f"Pred: {predicted_classes[i]}")  # Tahmin edilen sınıfı başlık olarak göster
    plt.axis('off')  # Eksenleri gizle
plt.show()
```

+ Code + Markdown

[9]:

```python
#veri setinin görselleştirilmesi
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Eğitim veri setini yükle
train_file = "/kaggle/input/data-set/train.csv"
train_data = pd.read_csv(train_file)

# Veriyi ön işleme
X = train_data.drop('label', axis=1).values  # Etiket hariç her şey (piksel değerleri)
y = train_data['label'].values  # Etiketler (0-9 arasındaki rakamlar)

# Veriyi normalize etme (piksel değerlerini 0-1 arasında)
X = X / 255.0

# Görüntüleri 28x28 boyutlarına dönüştürme ve yeniden şekillendirme
X = X.reshape(-1, 28, 28, 1)  # Her görüntü 28x28 boyutunda ve 1 kanal (grayscale)

# Görüntüleme: İlk 5 görüntüyü ve etiketlerini görselleştirme
plt.figure(figsize=(10, 10))
for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(X[i].reshape(28, 28), cmap='gray')
    plt.title(f"Label: {y[i]}")
    plt.axis('off')  # Eksenleri gizle
plt.show()
```



+ Code + Markdown

Veri setinin görselleştirilmesi

6. Gelecek Çalışmalar

- **Model İyileştirmeleri:** Konvolüsyonel katman sayısı artırılabilir, farklı aktivasyon fonksiyonları denenebilir.
- **Veri Artırma:** Veri artırma (data augmentation) teknikleri ile modelin genelleme başarısı artırılabilir.
- **Model Kaydetme ve Yükleme:** Eğitim tamamlandıktan sonra model kaydedilebilir ve daha sonra kullanılabilir.