T.C.

BALIKESİR ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

202013709082

Anıl Taha ADAK

BMM4101 YAPAY ZEKA TEKNİKLERİ BÜTÜNLEME ÖDEVİ

Danışman: Dr. Öğr. Üyesi Kadriye ERGÜN

BALIKESİR 02 – 2024

İÇİNDEKİLER:

ÖZET

Bu proje, yapay sinir ağları kullanarak el yazısı tanıma üzerine odaklanmaktadır. Tek bir yazı tipi içindeki 62 harfler(büyük/küçük) ve rakamlar) karakterin harf formlarını tanıtmak amacıyla gerçekleştirilen bu çalışmada, 3 katmanlı bir yapay sinir ağı modeli geliştirilmiştir.

Çalışmanın temel bileşenleri şunlardır:

Veri Seti: Çalışma, belirli bir yazı tipindeki 36 karakterin küçük ve büyük harf formlarını içeren bir veri setini temel almaktadır. Bu veri seti, el yazısı örneklerinin içerdiği harf ve sayı kombinasyonları ile zenginleştirilmiştir.

Ağ Topolojisi: Tek bir gizli katman içermekte olup, bu katman el yazısı karakterlerinin öğrenilmesinde temel rol oynamaktadır. Giriş katmanı, veri setindeki örneklerin özelliklerini içermekte ve çıkış katmanı, 62 farklı karakter sınıfını temsil etmektedir.

Eğitim ve Öğrenme: SGD, öğrenme sürecinde temel algoritma olarak kullanılmıştır. Model, eğitim veri seti üzerinde öğrenirken, doğrulama seti üzerindeki performansını değerlendirmek ve aşırı öğrenmeyi önlemek için gerekli düzenlemeler yapılmaktadır.

Sonuçlar ve Değerlendirme: Geliştirilen modelin performansı, doğrulama seti üzerinde elde edilen doğruluk oranları ve karışıklık matrisi gibi metriklerle değerlendirilmiştir. Elde edilen sonuçlar, modelin belirli bir yazı tipindeki el yazısı karakterlerini başarıyla tanıma yeteneğini göstermektedir.

Analiz ve Öneriler: Çalışma, modelin başarılarına ve zorluklarına odaklanarak, olası iyileştirmeler ve gelecekteki çalışmalar için öneriler sunmaktadır. Analiz, modelin genelleme yeteneği, öğrenme hızı ve performans üzerinde etkili olan faktörlere odaklanmaktadır.

Bu proje, yapay sinir ağları ile el yazısı tanıma konusunda başarılı bir örnek sunmaktadır. Modelin genel performansı, belirli bir yazı tipindeki karakterlerin tanınmasında etkili bir araç olarak gösterilmektedir.

ANAHTAR KELİMELER : Harf sınıflandırması, desen tanıma, Yapay sinir ağları, el yazısı tanıma

BÖLÜM 1.

GİRİŞ

Günümüzde el yazısı tanıma sistemleri birçok alanda kullanılmaktadır. Örneğin form veya dökümanların taranıp elektronik ortama aktarılmasında el yazısı tanıma sistemleri kullanılır. Bu sistemler özellikle geçmiş zamanlardan kalma dokümanların aktarımını büyük ölçüde kolaylaştırmaktadırlar. Günümüzde de özellikle bankalar gibi yazışmaların, bireysel işlemlerin yoğun olduğu iş alanlarında bilgi girişi maliyetini büyük oranda düşürebilmektedirler.

Bu örneklere ek olarak çevrimiçi(anlık) tanıma yapan sistemler de yaygın olarak kullanılmaktadır. Özellikle eğitim alanında bu yöntemlerden faydalanılmaktadır. Öğrencilerin yazmayı öğrenmesine katkı sağlayan çokça uygulama bulunmaktadır. Yine el yazısı tanıma tekniklerinden, bedensel veya zihinsel engeli olan kişiler için özelleştirilmiş eğitim-öğretim uygulamalarında yararlanılmaktadır.

Bu çalışmanın el yazısı tanıma alanında çalışmak isteyenler için, algoritma seçimi aşamasında bir kaynak teşkil etmesi amaçlanmıştır.

LİTERATÜR TARAMASI

Murat Şekerci, Türkçe el yazısı tanıma üzerine yaptığı çalışmada, karakter tanıma aşamasında korelasyon yöntemini kullanmış, K-en yakın komşuluk algoritması ile sistemlerini güçlendirmiştir. Ayrıca sözlük kullanılarak tanıma sistemi güçlendirilmiştir.[1]

Ayhan Erdem ve Emre Uzun, "Yapay Sinir Ağları İle Türkçe Times New Roman, Arial Ve El Yazısı Karakterleri Tanıma." yapılan bu çalışma, yapay sinir ağlarının el yazısı tanıma konusundaki potansiyelini göstermektedir. Geliştirilen algoritma ve program, karakter tanıma konusunda başarılı sonuçlar elde etmiş ve gerçek yaşam uygulamalarında kullanılabilecek bir potansiyele sahiptir.[4]

Orhan DURMAZ, Yapay Sinir Ağları ile Karakter Tanıma ile yapılan çalışmada verilen görsel üzerinde tanımlama yapmayı sağlamayı amaçlamış ve gerekli çıkarımlar yapılmıştır.[9]

KAYNAK ARAŞTIRILMASI

El yazısı tanıma sistemleri birçok alanda ihtiyaç duyulan sistemlerdir. Günümüzde hemen hemen tüm işlemler dijital ortama taşınmış durumdadır. Bu aşamada el yazısı tanıma sistemlerine büyük görev düşmektedir. Bu sistemlerle el yazısı ile yazılmış mevcut dokümanların dijital ortama aktarılması sağlanabilmektedir. Bunun dışında da eğitim, sağlık gibi birçok alanda el yazısı tanıma sistemleri büyük önem arz etmektedir.

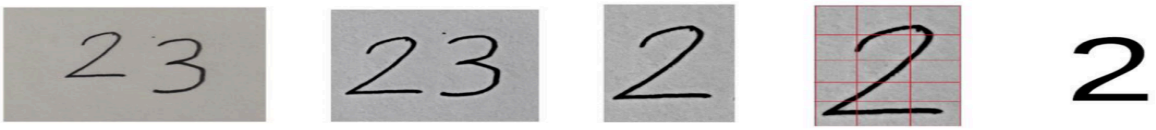
El yazısı tanıma yöntemleri iki grupta toplanabilir. Bunlar etkileşimsiz (çevrimdışı) ve etkileşimli (çevrimiçi) yöntemlerdir. Bizim kullanacağımız yöntem etkileşimsiz el yazısı tanımlamadır.[7]

Etkileşimli el yazısı tanıma

Etkileşimli yöntemler el yazısını yazıldığı sırada tanımaya yönelik tasarlanmışlardır. Genelde dokunmatik özelliği olan kalemler ve tabletlerden faydalanılarak kullanılırlar. Bu sistemlerde kalemin her hareketi, dokunuşu, devamlılığı izlenir.

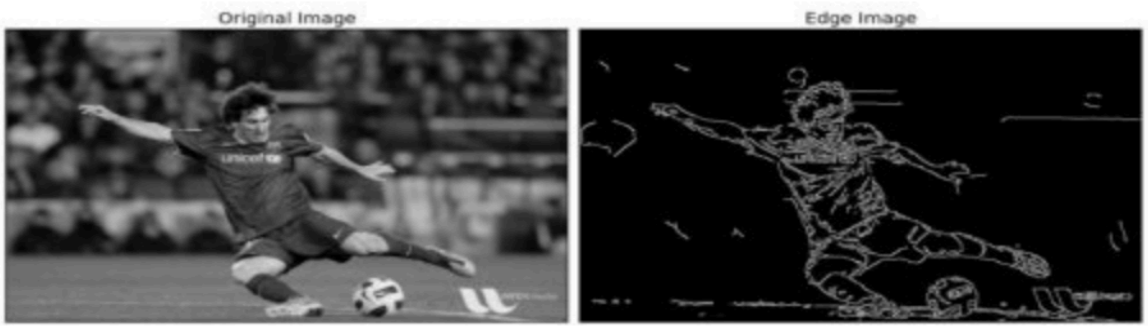
Etkileşimsiz el yazısı tanıma

Etkileşimsiz sistemler genelde önce kağıt üzerine el yazısının yazılması ve sonrasında dijital ortama aktarılması şeklindedir.



OPENCV-CANNY

OpenCV'deki Canny kenar tespiti yöntemi, bir görüntüdeki kenarları tespit etmek için kullanılan popüler bir algoritmadır. Bu algoritma, kenarlardaki belirginlikleri vurgular ve gürültüyü azaltmak için birçok aşamadan geçer. Çıktı olarak kontürleri çizmektedir.[10]



image

BÖLÜM 2.

AMAÇ

1. PROJENİN AMACI

Bu ödevin temel amacı, yapay sinir ağları (YSA) kullanarak belirli bir yazı tipindeki 62 karakteri (küçük ve büyük harflerle + rakamlar) tanımak ve tanıma işlemini gerçekleştirmektir. Yapay sinir ağları, bilgisayar sistemlerine insan benzeri öğrenme yeteneği kazandırmak için tasarlanmış bir yapılardır. Bu özel ödev, el yazısı karakterlerin dijital ortama girmesini sağlamak için bir yapay sinir ağı modeli geliştirmeyi amaçlamaktadır.

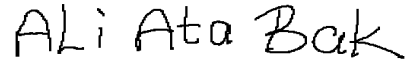
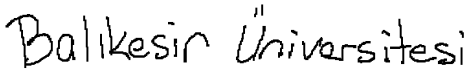
Yapay sinir ağları, genellikle büyük veri setlerinde öğrenme yeteneğine sahip olabilen ve genelleme yapabilen modeller oluşturabilen güçlü araçlardır. Bu projede, belirli bir yazı tipindeki 62 karakteri tanımak için bir YSA modeli kullanılacak. Modelin, el yazısı karakterleri doğru bir şekilde sınıflandırılabilmesi için eğitilmesi gerekmektedir.

El yazısı karakter tanıma için kullanılacak YSA modeli, öğrenme sürecini gerçekleştirmek ve modelin başarı oranını artırmak amacıyla uygun bir optimizasyon algoritması ve loss fonksiyonuyla derlenmelidir.

Bu projede ayrıca, el yazısı karakterlerin dijital ortama girmesini sağlamak için kullanılacak veri seti de büyük bir öneme sahiptir. Eğitim verilerinin çeşitliliği ve temsiliyeti, modelin gerçek dünya senaryolarında daha iyi performans göstermesine katkıda bulunacaktır.

Sonuç olarak, bu ödevde amaçlanan, belirli bir yazı tipindeki 36 karakterin YSA kullanılarak tanımlanmasıdır. Bu süreçte modelin eğitimi, optimizasyon ve doğru bir veri setinin kullanımı önemli adımları oluşturmaktadır.

Model üzerinden geçirilen el yazısı içeren png uzantılı dosyada bulunan karakterleri tahmin edip çıktısını vermesini amaçlanmıştır.

GİRDİ		ÇIKTI
	→	ALi Ata Bak
	→	Balıkesir Üniversitesi

2. PROJENİN KAPSAMI VE HEDEFLER

Projenin kapsamı ve hedefleri, yapay sinir ağları kullanarak el yazısı tanıma üzerine odaklanan bir öğrenme projesini içermektedir. Aşağıda proje kapsamı ve hedeflerini daha ayrıntılı bir şekilde açıklayabiliriz:

2.1. Projenin Kapsamı:

El Yazısı Veri Seti:

Kullanılacak olan el yazısı veri seti, içerisinde 36 karakteri içermektedir.

Veri seti, öğrenme sürecinde kullanılacak giriş verilerini ve bu verilere karşılık gelen etiketleri içermektedir.

Veri seti, standart OCR veri kümesi[8] ve kendi eklediğim yazı verileri ile elde edilmiştir.

Küçük harfler + büyük harfler + rakamlar = 62 karakter

Ağ Topolojisi:

Proje, örnek dosyada belirtilen özel bir ağ topolojisini kullanmaktadır

Giriş katmanında 784 nöron, 28*28 giriş boyutu ve sigmoid aktivasyonu içermektedir.

Gizli katman : 392 nöron ve sigmoid aktivasyonu içermektedir.

Çıkış katmanı: çıkış sayısı 62 nörondur ve softmax aktivasyonu içermektedir.

Ağ topolojisi, giriş katmanı, gizli katman ve çıkış katmanını içermektedir.

Öğrenme Yöntemi :

Proje kullanılan SGD, bir optimizasyon algoritmasıdır ve yapay sinir ağlarında yaygın olarak kullanılan bir öğrenme yöntemidir. Temel prensibi, modelin parametrelerini (ağırlıkları) optimize etmek ve belirli bir hedef (minimum veya maksimum) fonksiyonunu minimize etmektir. Ağırlıklar, her eğitim adımında gradyan (türev) kullanılarak güncellenir.

2.2. Projenin Hedefi

Projenin genel çerçevesini belirlerken, detaylı bir implementasyon ve matematiksel formülasyonlar, projenin gereksinimlerine bağlı olarak ayrıntılandırılabilir.

Veri Seti Ön İşleme:

Proje, kullanılacak el yazısı veri setini uygun formata getirecek ön işleme adımlarını içermelidir.

Giriş verileri ve etiketler belirlenmiş formatta hazırlanmalıdır.

Ağ Modelinin Oluşturulması:

Belirtilen ağ topolojisi kullanılarak ileri beslemeli geri yayılımlı yapay sinir ağı modeli oluşturulacaktır.

Giriş, gizli ve çıkış katmanları uygun sayıda nöron içerecektir.

Öğrenme Süreci:

SGD optimizasyon algoritması kullanılarak ağırlıklar güncellenir ve model belirli bir hedefe doğru ilerler. Eğitim süreci tamamlandığında, modelin test verileri üzerindeki performansı değerlendirilebilir.

Modelin Değerlendirilmesi:

Eğitilen model, ayrı bir test veri seti üzerinde değerlendirilecektir.

Doğruluk (accuracy) ve diğer ilgili metrikler kullanılarak modelin performansı ölçülecektir.

Tahminlerin Yapılması:

Eğitilen model, gerçek dünyadan el yazısı örnekleri üzerinde tahminler yapabilecek şekilde kullanılacaktır.

Elde edilen çıktılar kullanıcının okunabilir bir formatta sunulacaktır.

BÖLÜM 3.

YÖNTEM

Projenin yöntemi, el yazısı tanıma üzerine odaklanan yapay sinir ağı modelinin oluşturulması, eğitilmesi ve değerlendirilmesini içerir. Aşağıda projenin temel yöntemi adım adım gösterilmiş ve adımlar ilgili başlık altında ayrıntılı olarak açıklanmıştır.

1.VERİ KÜMESİ

1.1 VERİ SETİ HAZIRLIĞI

Kullanılacak olan el yazısı veri seti, belirlenen 62 karakteri içermektedir. El yazısı örnekleri küçük ve büyük harflerle temsil edilmektedir.. Veri seti, hazır olarak standart OCR veri kümesi ve elde edilen diğer veriler ile oluşturulmuştur. Bu kümede “A-Z” ve “0-9” aralıklarında bulunan harflerin standart sağlanmış veri setidir. data klasörü içerisinde sınıflandırılmıştır.

1.2 VERİ SETİ ÖN İŞLEME

Veri setindeki örnekler dosya adı(etiket) altında np dizisine alınarak işlem yapılmak üzere tutulmaktadır.

```
# Veri setini yükleme ve hazırlama
data_path = '/kaggle/input/standard-ocr-dataset/data/training_data'
karakterler = os.listdir(data_path)

harf_resim = []
etiketler = []

for karakter in karakterler:
    char_path = os.path.join(data_path, karakter)

    for file in os.listdir(char_path):
        img_path = os.path.join(char_path, file)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (28, 28))  # Giriş boyutuna uygun olarak yeniden boyutlandırma
        img = np.reshape(img, (28 * 28))  # Düzleştirme
        harf_resim.append(img)
        etiketler.append(karakter)
```

```
X = np.array(harf_resim)
y = np.array(etiketler)

print("X şekil:", X.shape)
print("y şekil:", y.shape)

# X ve y'nin içeriğini yazdırma (örneğin, ilk öğe)
print("X:", X[0])
print("y:", y[0])
```

Örnek olarak ilk elemanın NP'de tutulan formatı:

```
X şekil: (20628, 784)
y şekil: (20628,)
X: [220 219 220 217 214 213 215 219 220 218 219 219 222 221 219 220 221 220
 222 220 222 220 218 215 213 213 216 219 222 216 202 176 158 158 177 202
 220 222 224 222 223 221 220 221 221 219 218 221 223 219 207 186 165 164
 180 203 221 208 166 106  74  74 105 156 201 217 220 222 223 221 224 222
 221 222 222 222 220 206 168 118  80  78 119 182 219 205 155  83  42  36
  53  97 158 202 215 218 223 222 223 221 221 218 223 222 218 195 136  69
  42  52 100 175 221 207 158  86  42  29  34  52 100 163 202 216 223 222
 220 222 223 223 221 220 218 189 121  60  37  49 108 183 222 206 161  89
  43  30  28  34  52 104 164 202 217 220 221 223 220 221 222 220 218 190
 125  60  39  56 111 181 218 208 162  91  44  30  29  30  31  50 103 169
 206 218 220 219 224 221 221 223 218 194 129  62  37  57 116 185 219 209
 159  92  46  31  31  28  28  34  58 111 168 204 216 220 224 219 222 222
 218 197 138  69  40  55 114 184 224 210 166  92  46  41  47  38  29  32
  34  59 111 172 210 218 222 223 222 219 216 200 141  72  39  56 117 183
 219 207 160  92  50  57  79  72  42  28  29  36  61 115 173 207 219 221
 222 223 219 201 149  77  43  58 117 182 221 208 165  93  51  66 111 115
  72  36  30  29  33  63 116 173 207 218 219 223 217 202 152  83  46  57
 118 184 221 209 162  93  51  68 130 158 117  64  35  27  31  38  68 120
 179 212 218 221 220 201 159  89  46  56 118 184 221 207 162  93  52  69
 131 181 170 106  55  32  29  30  38  70 122 180 212 218 221 206 164  94
  50  59 117 183 220 208 164  92  47  63 131 189 199 164 100  54  34  28
  31  40  71 128 183 211 217 208 170 101  52  58 114 185 220 208 164  93
  52  63 126 187 213 199 158  92  47  31  29  30  38  71 131 186 211 210
 176 106  54  61 118 186 220 209 163  94  49  58 117 186 217 215 195 145
  86  47  32  28  31  39  79 138 188 208 182 112  57  62 114 182 221 207
 162  92  49  56 113 182 215 221 216 191 143  83  44  30  29  30  42  82
 146 189 180 119  58  60 116 184 217 210 165  93  50  54 106 175 215 222
 220 213 189 137  78  46  33  29  29  46  90 149 168 117  60  60 118 185
 222 208 164  93  49  51 101 171 213 220 221 219 213 186 133  77  44  32
  29  32  50  97 135 113  65  60 116 186 219 207 163  93  47  51 101 169
 212 220 222 222 219 211 185 130  75  37  27  30  34  54  83  88  60  64
 116 187 219 208 164  95  49  47  92 163 209 220 222 223 222 219 212 185
 131  74  38  31  28  33  47  55  47  60 117 186 226 210 166  95  48  47
  87 163 209 218 221 221 222 221 219 213 186 134  75  40  33  29  32  33
  37  58 117 186 218 209 163  92  45  43  82 157 208 221 222 221 221 221
 219 218 210 185 134  78  45  31  29  29  35  57 117 185 221 206 161  86
  43  40  79 151 206 220 222 222 222 220 221 222 221 215 190 137  82  47
  31  30  33  56 115 183 219 200 150  77  40  41  79 153 204 219 219 221
 222 222 221 221 223 219 212 193 139  83  48  33  32  52 113 181 217 199
 139  68  46  56  96 165 210 216 220 219 221 221 221 221 221 224 219 216
 190 144  89  53  41  55 111 180 220 200 159 113  97 113 155 199 219 222
 219 222 220 221 222 222 222 221 223 220 215 195 156 115  95 105 144 193
 220 215 204 188 184 191 206 216 222 221 221 222 222 220 222 221 218 220
 221 223 221 217 208 191 183 185 199 212]
y: N
```

Etiketleri encode etme ile sayısal değere dönüştürme:

```
# Etiketleri LabelEncoder kullanarak encode etme
# LabelEncoder'ı oluştur
le = LabelEncoder()

# Etiketleri sayısal değerlere dönüştür
y_encoded = le.fit_transform(y)
```

Veri setini karıştırmak,veri setindeki örneklerin sırasını rastgele değiştirerek modelin öğrenme sürecini iyileştirebilir. Bunu için shuffle fonksiyonu kullanılmıştır.

```
# Veriyi karıştırma
X_sh, y_sh = shuffle(X, y, random_state=42)
```

Burada X ve y sırasıyla özellik ve etiket verilerini temsil eder. random_state parametresi, her seferinde aynı karıştırma sırasını elde etmek için kullanılır. Eğer bu parametre verilmezse, her çağrıldığında farklı bir karıştırma elde edilmektedir.

2. MODEL MİMARİSİ

Bu projede oluşturulan model, üç katmandan oluşan basit bir yapısı olan bir yapay sinir ağıdır. İşte katmanlar ve işlevleri:

```
# Veriyi eğitim ve doğrulama setlerine ayırma
X_train, X_val, y_train, y_val = train_test_split(X_sh, y_sh, test_size=0.2, random_state=42)

# Modeli oluşturma
model = Sequential()
model.add(Dense(units=784, activation='sigmoid', input_dim=28 * 28))  # Giriş boyutu 28 * 28
model.add(Dense(units=392, activation='sigmoid'))
model.add(Dense(units=36, activation='softmax'))  # Çıkış sınıf sayısı 36

# Modeli derleme
model.compile(optimizer=SGD(learning_rate=0.02), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Giriş Katmanı (Dense):

- units=784: Bu katman 784 nörona sahiptir, çünkü giriş boyutu 28x28 piksel resimlerden gelmektedir (28 * 28 = 784).
- activation='sigmoid': Aktivasyon fonksiyonu olarak sigmoid kullanılmıştır.

Gizli Katman (Dense):

- units=392: Bu gizli katman 392 nörona sahiptir.
- activation='sigmoid': Sigmoid aktivasyon fonksiyonu kullanılmıştır.

Çıkış Katmanı (Dense):

- units=62: Bu çıkış katmanı 36 nörona sahiptir. Çünkü, varsayılan olarak softmax aktivasyonu kullanıldığında, çıkış katmanındaki her nöron, farklı bir sınıfı temsil eder.
- activation='softmax': Çıkış katmanında softmax aktivasyon fonksiyonu kullanılmıştır.

Bu model, genellikle 28x28 piksel boyutundaki görüntülerle yapılan bir sınıflandırma görevi için kullanılabilir. Sigmoid aktivasyonları ve softmax, özellikle çoklu sınıflandırma görevlerinde yaygın olarak kullanılan aktivasyon fonksiyonlarıdır. Bu model, eğitim sırasında stokastik gradyan iniş (SGD) optimizasyon algoritması ve sparse kategorik çapraz entropi kaybını kullanarak eğitilmektedir.

2.1 YSA MODELİNİN DERLENMESİ

optimizer=SGD(learning_rate=0.02): Optimizasyon algoritması olarak Stochastic Gradient Descent (SGD) kullanılmaktadır. SGD, model parametrelerini güncellerken her seferinde yalnızca bir alt küme örnek (mini-batch) kullanarak gradyan inişi gerçekleştirir. learning_rate parametresi, öğrenme oranını belirler. Bu oran, her güncelleme adımında model parametrelerinin ne kadar değiştirileceğini kontrol eder.

loss='sparse_categorical_crossentropy': Kayıp fonksiyonu olarak sparse categorical crossentropy kullanılır. Bu, sınıflandırma problemleri için yaygın bir kayıp fonksiyonudur. sparse_categorical_crossentropy, hedef etiketlerin tamsayı formatında olduğu durumlar için uygundur.

metrics=['accuracy']: Modelin değerlendirilmesi için kullanılan metrikler belirlenir. Burada sadece doğruluk (accuracy) metriği kullanılmaktadır. Doğruluk, modelin doğru sınıflandırma yüzdesini ölçer. Eğitim sırasında bu metriklerin izlenmesi, modelin performansının takip edilmesine yardımcı olmaktadır..

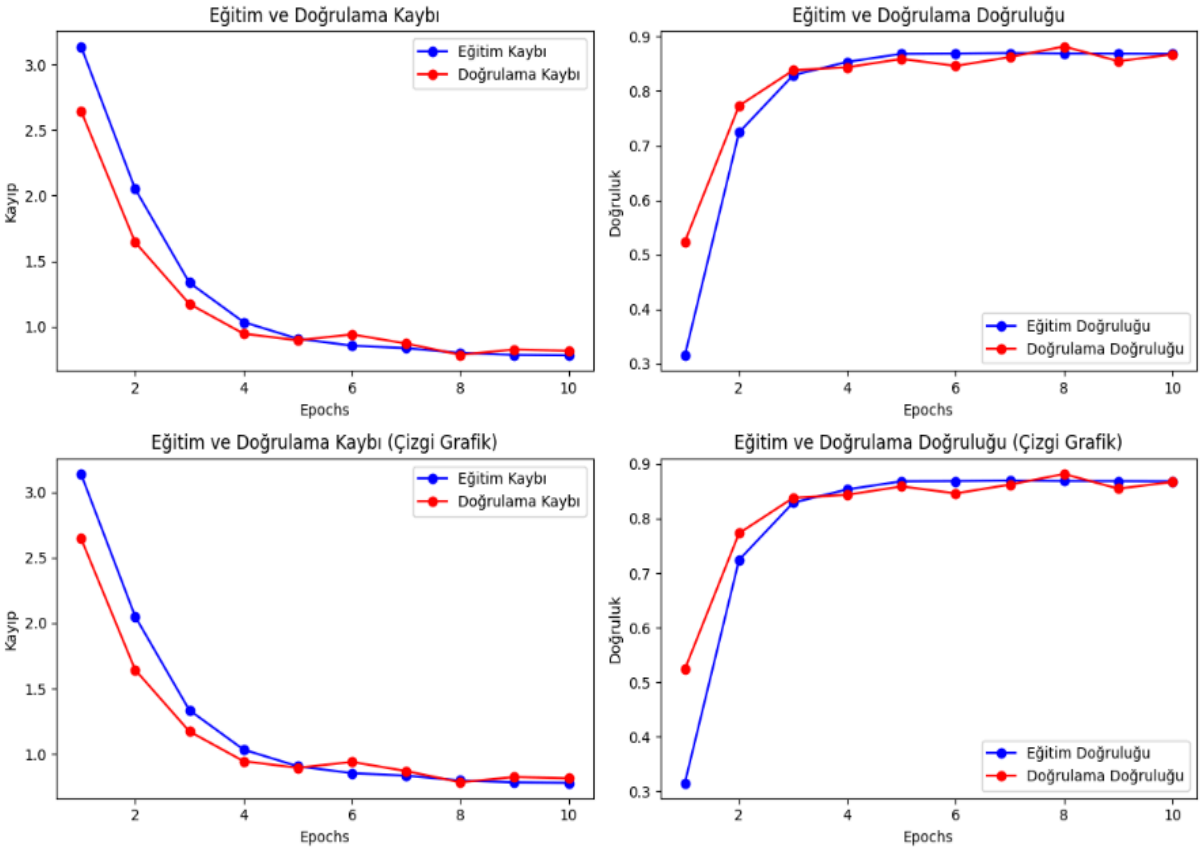
3. EĞİTİM

Model eğitimi history değişkenine atanarak eğitim performansı görselleştirilmiştir.

```
# Modeli eğitme
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=10, batch_size=32)
```

```
Epoch 1/10
516/516 [==============================] - 2s 3ms/step - loss: 3.1834 - accuracy: 0.2937 - val_loss: 2.7222 - val_accuracy: 0.5548
Epoch 2/10
516/516 [==============================] - 2s 3ms/step - loss: 2.1207 - accuracy: 0.7074 - val_loss: 1.6961 - val_accuracy: 0.7887
Epoch 3/10
516/516 [==============================] - 2s 3ms/step - loss: 1.3613 - accuracy: 0.8260 - val_loss: 1.1705 - val_accuracy: 0.8260
Epoch 4/10
516/516 [==============================] - 2s 3ms/step - loss: 1.0388 - accuracy: 0.8587 - val_loss: 0.9899 - val_accuracy: 0.8379
Epoch 5/10
516/516 [==============================] - 2s 3ms/step - loss: 0.9235 - accuracy: 0.8652 - val_loss: 0.9817 - val_accuracy: 0.8536
Epoch 6/10
516/516 [==============================] - 2s 3ms/step - loss: 0.8616 - accuracy: 0.8715 - val_loss: 0.8437 - val_accuracy: 0.8657
Epoch 7/10
516/516 [==============================] - 2s 3ms/step - loss: 0.8155 - accuracy: 0.8725 - val_loss: 0.7947 - val_accuracy: 0.8815
Epoch 8/10
516/516 [==============================] - 2s 3ms/step - loss: 0.7901 - accuracy: 0.8737 - val_loss: 0.9380 - val_accuracy: 0.8313
Epoch 9/10
516/516 [==============================] - 2s 3ms/step - loss: 0.8172 - accuracy: 0.8708 - val_loss: 0.8299 - val_accuracy: 0.8730
Epoch 10/10
516/516 [==============================] - 2s 3ms/step - loss: 0.8133 - accuracy: 0.8659 - val_loss: 0.8349 - val_accuracy: 0.8623
```

Görselleştirme için Chatgpt'ten faydalanılmıştır.



4. TEST EDİLMESİ

İçeriği el yazısı olan png dosyası OpenCv ile karakterlere ayrılarak yeni bir klasör içerisine tek tek oluşturulur. ve oluşan yeni resimler sırasıyla modelden

geçirilip, hangi kategoride ise o kategorinin etiketini alır ve diziye kaydedilir. Klasör içerisinde resimleri okuma işlemi tamamlandıktan sonra dizide tutulan etiket isimleri sırasıyla yazdırılır ve ilk verilen png içerisinde ki el yazısı içeriği dizi içerisinde tutulmuştur. ve dizi ekrana yazdırılarak işlem tamamlanmaktadır.

Karakterleri bulma ve ayırma işleminde Opencv'de bulunan Canny algoritması kullanılmıştır.

png dosyasının karakterlere ayrılması:

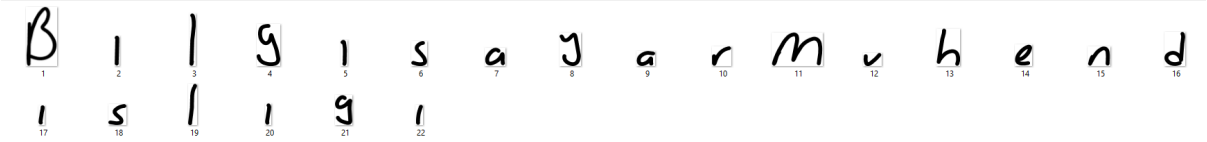
```
import cv2
import os

png_path=
# Görüntüyü yükle
image = cv2.imread("./3.png")

# Görüntüyü siyah beyaz yap
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Canny kenar tespiti yap
edges = cv2.Canny(gray_image, threshold1=30, threshold2=100)

# Kenarlar arasında kontur bul
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Konturları çizmek için kopya bir görüntü oluştur
contour_image = image.copy()
# Her kontur için işlem yap
for contour in contours:
    # Konturu çevreleyen bir dikdörtgen al
    x, y, w, h = cv2.boundingRect(contour)
    # Konturu çevreleyen dikdörtgeni kırp
    character = image[y:y+h, x:x+w]
    # Karakteri ayrı bir dosyada kaydet
    if y+h-y>15 and x+w-x>3:
        cv2.imwrite(f"f/character_{x}_{y}.png", character)

# Görüntüyü göster
cv2.imwrite("Edges.jpg", edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Örnek olarak bir png'in bu işleme girmesi ve çıktısı:

Verilen png:

Kontur çizimi:

Karakterlerin ayrılması ve klasör içerisine kaydedilmesi sonucu çıktı:



Elde edilen resimlerin sırayla model üzerinden tahmin edilmesi ve etiketlerin dizide tutulup çıktı olarak verilmesi:

```
import cv2
import os
import numpy as np
from keras.models import load_model

# Önceden eğitilmiş modelin yüklenmesi
model = load_model('your_model.h5')  # Model dosyanızın adını belirtin

# Klasör yolunu belirtin
folder_path = '/kaggle/output/png-elyazisi'

# PNG dosyalarını okuma ve model tahminleri için kullanma
predictions = []

for filename in os.listdir(folder_path):
    if filename.endswith(".png"):
        # PNG dosyasını oku
        img_path = os.path.join(folder_path, filename)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (28, 28))  # Giriş boyutuna uygun olarak yeniden boyutlandırma
        img = np.reshape(img, (28 * 28))  # Düzleştirme

        # Görüntüyü modele sokma ve tahmin yapma
        img = np.reshape(img, (1, your_image_width, your_image_height, 1))  # Giriş boyutunu düzenle
        prediction = model.predict(img)

        # Tahmin sonuçlarını listeye ekle
        predictions.append(prediction)

# Tahmin sonuçlarını numpy dizisine dönüştürme
predictions = np.array(predictions)

# Elde edilen tahmin sonuçlarını kullanarak gerekli işlemleri yapabilirsiniz.
# Örneğin, sınıflandırma yapıyorsanız, argmax kullanarak sınıf indekslerini alabilirsiniz.
predicted_classes = np.argmax(predictions, axis=1)

# Artık `predicted_classes` dizisinde her bir görüntü için model tahminlerini bulabilirsiniz.
print(predicted_classes)
```

Parçalanan karakterler tek tek aynı yapıya çekilerek model üzerinden tahmin işlemi gerçekleştirilmekte ve modelden dönen etiket değeri diziye aktarılmaktadır.

5. SONUÇLAR

Projeni sonuçlarını çıktı üzerinden değerlendirilmesi:

PNG olarak verilen dosya içeriği:

Çıktısı:

```
# Artık `predicted_classes` dizisinde her bir görüntü için model tahmi
print(predicted_classes)
```

```
Biigisayarmuhendisigi
```

PNG olarak verilen dosya içeriği:

Çıktı:

```
# Artık `predicted_classes` dizisinde her bir görüntü için mo
print(predicted_classes)
```

```
Ani1TahaAdaK
```

PNG olarak verilen dosya içeriği:

Çıktı:

```
# Artık `predicted_classes` dizisinde her bir görünt
print(predicted_classes)
```

```
Balike5irUniulr5itesi
```

6. ANALİZ

Proje dahilinde istenilen işlemler gerçekleştirildi. Modelin eğitilmesi sonucu elde edilen sonuçlar durumunun analiz edilmesi

Proje dahilinde yapılan işlemler:

- El yazısı içerikleri modelde kullanılmak üzere gerekli ayarlamaları yapıldı.
    - El yazısı içeren resimler aynı boyuta dönüştürülmektedir.
    - Resimler numpy ile çok boyutlu(28x28) matrise alınmaktadır.
- Etiketler sayısal formata dönüştürülmektedir.
- Model girecek veriler Suffle fonksiyonu ile karıştırılarak eğitimin daha başarılı olması sağlanmıştır.
- Model istenilen mimaride oluşturulmaktadır.
- Model üzerinden tahminde bulunulacak el yazısı içeren resim istenilen formata çevrilmekte ve numpy dizisine alınmaktadır.
- Verilen el yazısı içeriğinin tahmin sonucu çıktı olarak alınmaktadır.

Projenin analizi:

Proje genel anlamda istenileni yapmaktadır. Png ile verilen el yazısı içeriğini genel olarak doğru tahmin etmektedir.

Tahmin de karıştırılan karakterler yazılmasına bağlı olarak doğru tahmin edilmesi değişmektedir. Örnek olarak çok fazla karışan (I,İ,1,l),(S,5),(O,Ö,0) karakterleri yazılmasına bağlı olarak tahmin sonucu verilen etiket değeri değişim göstermektedir.

Sonucun daha iyi doğruluk vermesi ve performansın artması için yapılması gerekenler:

- Veri setinin genişletilmesi ve daha fazla benzersiz yazı stili içeriği eklenmesi
- Boşluk karakterinin daha iyi tanıtılması ve tahmin için verilen resimde boşlukların da ayırt edilmesini sağlamak
- Modelin katmanları, nöronları ve yapı mimarisinin farklı kombinasyonlarında denenerek eğitim yapılmasını sağlamak
- Png olarak verilen el yazısı içeriğinin karakterlere parçalarken farklı yapılarda parçalanması ile karakterlerin daha iyi ayırt edilmesini sağlanması

7. TAM KOD

Verinin yüklenmesi ve hazırlanması

```
def load_and_prepare_data(data_path):
    karakterler_resim = []
    karakterler_etiket = []

    characters = os.listdir(data_path)

    for char in characters:
        char_path = os.path.join(data_path, char)

        for file in os.listdir(char_path):
            img_path = os.path.join(char_path, file)
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            img = cv2.resize(img, (28, 28))
            img = np.reshape(img, (28 * 28))
            karakterler_resim.append(img)
            karakterler_etiket.append(char)

    X = np.array(karakterler_resim)
    y = np.array(karakterler_etiket)

    # Etiketleri LabelEncoder kullanarak encode etme
    label_encoder = LabelEncoder()
    y_encoded = label_encoder.fit_transform(y)

    # Etiketleri one-hot encoding'e dönüştürme
    y_encoded = to_categorical(y_encoded, num_classes=len(np.unique(y)))

    # Veriyi karıştırma
    X_shuffle, y_shuffle = shuffle(X, y_encoded, random_state=36)

    # Veriyi eğitim ve doğrulama setlerine ayırma
    X_train, X_val, y_train, y_val = train_test_split(X_shuffle, y_shuffle, test_size=0.2, random_state=42)

    return X_train, X_val, y_train, y_val, le
```

Eğitim modelinin oluşturulması

```
def create_and_train_model(X_train, y_train, X_val, y_val):
    model = Sequential()
    model.add(Dense(units=784, activation='sigmoid', input_dim=28 * 28))
    model.add(Dense(units=392, activation='sigmoid'))
    model.add(Dense(units=36, activation='softmax'))

    model.compile(optimizer=SGD(learning_rate=0.02), loss='categorical_crossentropy', metrics=['accuracy'])

    history = model.fit(X_train, y_train, validation_data=(X_val, y_val), batch_size=32, epochs=10)

    return model, history
```

Modelin görselleştirilmesi

```
def plot_training_history(history):
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 3, 1)
    plt.plot(history.history['loss'], label='Eğitim Kaybı')
    plt.plot(history.history['val_loss'], label='Doğrulama Kaybı')
    plt.xlabel('Epochs')
    plt.ylabel('Kayıp')
    plt.title('Kayıp Grafiği')
    plt.legend()

    plt.subplot(1, 3, 2)
    plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')
    plt.plot(history.history['val_accuracy'], label='Doğrulama Doğruluğu')
    plt.xlabel('Epochs')
    plt.ylabel('Doğruluk')
    plt.title('Doğruluk Grafiği')
    plt.legend()

    plt.subplot(1, 3, 3)
    plt.plot(history.history['val_loss'], label='Doğrulama Kaybı')
    plt.xlabel('Epochs')
    plt.ylabel('Doğrulama Kaybı')
    plt.title('Doğrulama Kaybı Grafiği')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

Model kaydetme fonksiyonu

```
def save_model(model, model_path='model.h5'):
    model.save(model_path)
```

Tahmin edilecek png dosyasının yüklenmesi ve işlenmesi

```
def load_and_process_image(image_path, contours_data="a"):
    image = cv2.imread(image_path)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray_image, threshold1=30, threshold2=100)
    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    i = 1
    folder_path = os.path.join(os.getcwd(), contours_data)
    os.makedirs(folder_path, exist_ok=True)

    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        character = image[y:y + h, x:x + w]

        if y + h - y > 15 and x + w - x > 3:
            cv2.imwrite(os.path.join(folder_path, f"{i}.png"), character)
            i += 1

    cv2.imwrite("Edges.jpg", edges)
```

Model üzerinde tahmin yapma işleminin yapılma işlemi, tahmin edilmesi istenilen png dosyasının uygun format dahilinde modelden geçirilmesi ve tahmin saonucun verdiği etiketin dönüştürülerek yazdırılması

```python
def predict_with_model(model, folder_path):
    predictions = []

    for filename in sorted(os.listdir(folder_path)):
        if filename.endswith(".png"):
            img_path = os.path.join(folder_path, filename)
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            img = cv2.resize(img, (28, 28))
            img = np.reshape(img, (28 * 28))
            img = np.reshape(img, (1, 784, 1))
            img = img / 255.0
            prediction = model.predict(img)
            predictions.append(prediction)

    predictions = np.array(predictions)
    predicted_classes = np.argmax(predictions, axis=1)

    return predictions  # Return the predictions list

# Veri setini yükleme ve modeli eğitme
X_train, X_val, y_train, y_val, label_encoder = load_and_prepare_data('/kaggle/input/data')
model, history = create_and_train_model(X_train, y_train, X_val, y_val)

# Eğitim sürecini görselleştirme
plot_training_history(history)

# Modeli kaydetme
save_model(model, 'model.h5')

# El yazısı içeren görüntüyü işleme ve tahmin
load_and_process_image("/kaggle/input/png-elyazisi/yz/1.jpg")
predictions = predict_with_model(model, 'a')  # Use the returned predictions

# Tahmin edilen sınıfları etiketlere dönüştürme
predicted_classes = np.argmax(predictions, axis=2)
predicted_labels = label_encoder.inverse_transform(predicted_classes)
print(predicted_labels)
```

BÖLÜM 4.

SONUÇ

Proje dahilinde yapılan çalışma sonucunda, kullanıcı girdisi olan el yazısı bulunan png dosyasının içeriğini okumayı sağlandı.

Oluşturulan YSA modelinin ağ topolojisinde öğrenme yöntemi olarak SGD (Stochastic Gradient Descent) algoritması ve tek gizli katmanlı ileri beslemeli geri yayılımlı YSA modelini kullandı. 1 giriş + 1 gizli +1 çıkış katmanı kullanıldı. Oluşturulan ANN modeli ile verilen el yazısı png dosyası harflere ayırarak her harf modelden geçirilmektedir. ve sınıflandırma sonucu elde edilen etiketler sırasıyla diziye eklenmektedir. İşlem sonucu dizi ekranda gösterilmektedir.

Proje sayesinde Model eğitimi, Eğitim sonucu değerlendirme, Model kullanımı ve resim verilerinin işlem için uyarlanması yetenekleri elde edilmiştir.

Yapılan çalışmalar ile el yazısı tanıma alanında araştırmalar yapılmış ve bu alanda kaynak olabilecek bir çalışma hazırlanmıştır.

Yapılan çalışmalar ile dijital ortamda olmayan kullanıcı girdilerinin fotoğrafları sayesinde dijital ortama aktarımını sağlayarak, tüm verilerin tek bir yerde ve düzenli tutulmasını sağlanabilir.

Veri seti farklı dil seçenekleri ile artırılarak, tercümanlık, arkeoloji, tarih, sağlık, askeri ve sanat alanları gibi alanlarda kullanılabilir.

KAYNAKLAR

[1] Şekerci, M. (2007). Birleşik ve eğik Türkçe el yazısı tanıma sistemi. Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi.

[2] Şekerci, M., & Kandemir, R. (2006). Sözlük kullanarak Türkçe el yazısı tanıma. Edirne.

[3] Yılmaz, B. (2014). Öğrenme güçlüğü çeken çocuklar için el yazısı tanıma ile öğrenmeyi kolaylaştırıcı bir mobil öğrenme uygulaması tasarımı. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi.

[4] Erdem, O. A., & Uzun, E. (Bilinmeyen tarih). Yapay Sinir Ağları İle Türkçe Times New Roman, Arial Ve El Yazisi Karakterleri Tanima. Elektronik Bilgisayar Eğitimi Bölümü, Teknik Eğitim Fakültesi, Gazi Üniversitesi.

[5] Tosun, S. (2007). Sınıflandırmada yapay sinir ağları ve karar ağaçları karşılaştırması: Öğrenci başarıları üzerine bir uygulama. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Bölümü, Yüksek Lisans Tezi.

[6] Güzel, Y. (Bilinmeyen tarih). Çok Katmanlı Yapay Sinir Ağı. https://medium.com/@yasinguzel/yapay-zeka-ders-notlar

[7] Karakaya, R. (Bilinmeyen tarih). MAKİNE ÖĞRENMESİ YÖNTEMLERİYLE EL YAZISI TANIMA.

[8] Preatcher. (Bilinmeyen tarih). Standart OCR veri kümesi. https://www.kaggle.com/datasets/preatcher/standard-ocr-dataset

[9] Durmaz, O. (Bilinmeyen tarih). Yapay Sinir Ağları ile Karakter Tanıma. www.osmandurmaz.com

[10] OpenCV. (Bilinmeyen tarih). Canny algoritması dökümantasyonu. https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html

[11] OpenAI. (Bilinmeyen tarih). ChatGPT: OpenAI'nin kod düzenleme ve metin oluşturmaya yönelik dil modeli.