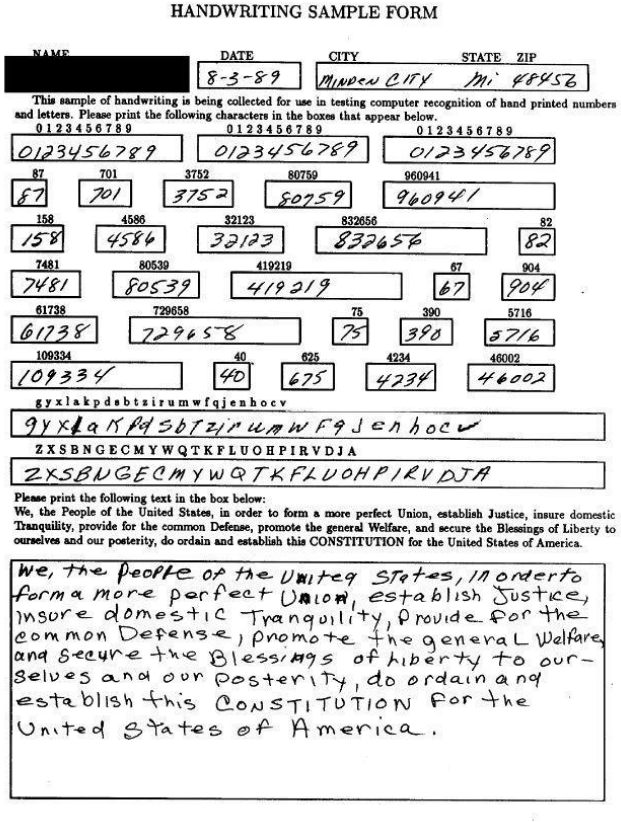
İÇİNDEKİLER

1. MNIST VERİ SETİ

MNIST veri tabanı (Modified National Institute of Standards and Technology database), çeşitli görüntü işleme sistemlerini eğitmek için yaygın olarak kullanılan, elle yazılmış büyük bir veri tabanıdır. Veri tabanı ayrıca makine öğrenimi alanında eğitim ve test için yaygın olarak kullanılmaktadır. NIST'in orijinal veri kümelerinden alınan örneklerin "yeniden karıştırılması" ile oluşturulmuştur. İçerik oluşturucular, NIST'in eğitim veri kümesi Amerikan Sayım Bürosu çalışanlarından, test veri kümesi Amerikan lise öğrencilerinden almışlardır.


HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789 0123456789 0123456789

0123456789 0123456789 0123456789

87 701 3752 80759 960941

87 701 3752 80759 960941

158 4586 32123 832656 82

158 4586 32123 832656 82

7481 80539 419219 67 904

7481 80539 419219 67 904

61738 729658 75 390 5716

61738 729658 75 390 5716

109334 40 625 4234 46002

109334 40 675 4234 46002

gyxlakpdsbtsirumwfqjenhocv

gyxlakpdsbtsirumwfqjenhocv

ZXSBNGECMYWQTKFLUOHPIRVDJA

ZXSBNGECMYWQTKFLUOHPIRVDJA

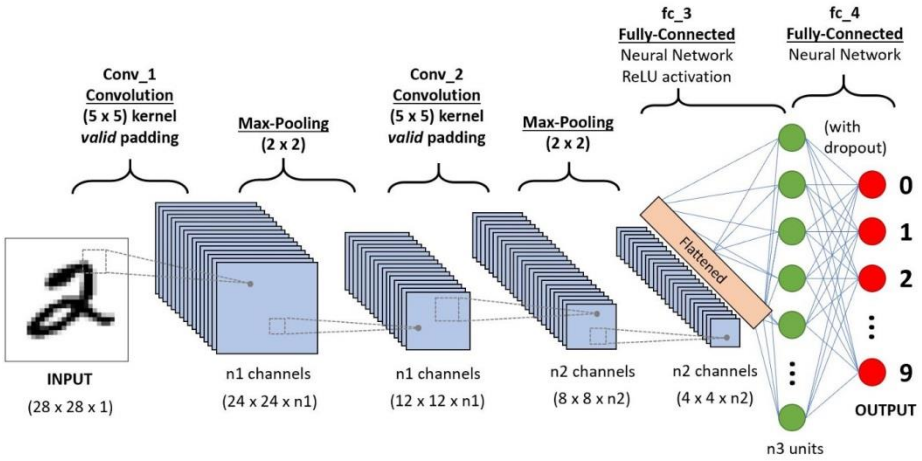
Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, Provide for the common Defense, promote the general Welfare, and secure the Blessings of liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

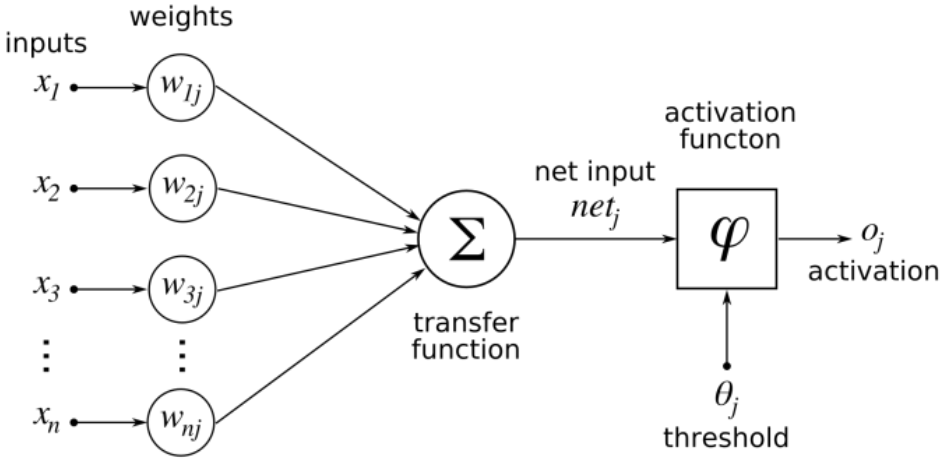
Alınan görüntüler 28x28 çözünürlüğe ve gri tonlama düzeylerini getirilerek kenar yumuşatma uygulanmıştır. MNIST veri seti 60.000 eğitim görüntüsü ve 10.000 test görüntüsü içermektedir.

2. CNN (Convolutional Neural Network)

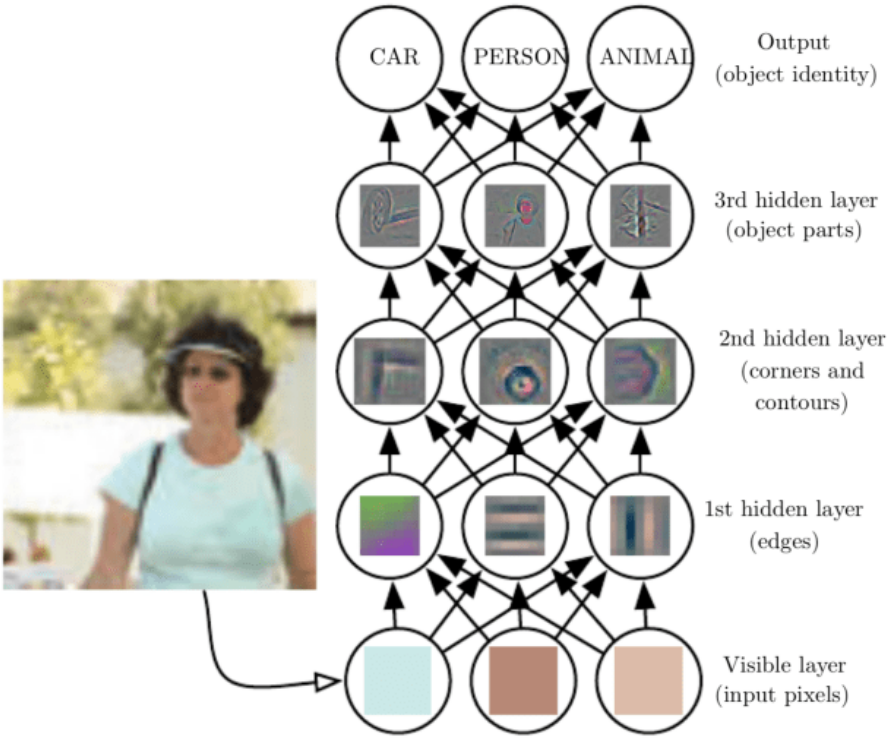
Evrişimsel sinir ağları, derin öğrenmenin bir alt dalıdır ve genellikle görsel bilginin analiz edilmesinde kullanılır. Yaygın kullanım alanları resim ve video tanıma, öneri sistemleri, resim sınıflandırma, tıbbi görüntü analizi ve doğal dil işleme olarak sıralanabilir. Konvolüsyon (evrişim) özel bir doğrusal (lineer) işlem türüdür. Evrişimli sinir ağları, katmanlarından en az birinde genel matris çarpımı yerine evrişimi kullanan basit sinir ağlarıdır.



Evrişimsel sinir ağları, çoklu yapay nöron katmanlarından oluşur. Biyolojik muadillerinin kaba bir taklidi olan yapay nöronlar, çoklu girdilerin ağırlıklı toplamını hesaplayan ve bir aktivasyon değeri veren matematiksel fonksiyonlardır.



Her nöronun davranışı ağırlıkları ile tanımlanır. Bir CNN'nin yapay nöronları piksel değerleriyle beslendiğinde çeşitli görsel özellikler seçer. ConvNet'e bir görüntü girdiğinizde, katmanlarının her biri birkaç aktivasyon haritası oluşturur. Aktivasyon haritaları, görüntünün ilgili özelliklerini vurgular. Nöronların her biri girdi olarak bir piksel parçası alır, renk değerlerini ağırlıklarıyla çarpar, toplar ve aktivasyon fonksiyonu boyunca çalıştırır.



3. KULLANILAN TEKNOLOJİLER

3.1 Python

Nesne yönelimli, yorumlamalı, birimsel (modüler) ve etkileşimli yüksek seviyeli bir programlama dilidir. Girintilere dayalı basit söz dizimi, dilin öğrenilmesini ve akılda kalmasını kolaylaştırır. Bu da ona söz diziminin ayrıntıları ile vakit yitirmeden programlama yapılmaya başlanabilen bir dil olma özelliği kazandırır.

3.2 TensorFlow

TensorFlow, makine öğrenimi için ücretsiz ve açık kaynaklı bir yazılım kütüphanesidir. Bir dizi görevde kullanılabilir, ancak derin sinir ağlarının eğitimi ve çıkarımına özel olarak odaklanmaktadır. Tensorflow, veri akışına ve türevlenebilir programlamaya dayalı sembolik bir matematik kitaplığıdır. Google'da hem araştırma hem de üretim için kullanılmaktadır. TensorFlow, Google Brain ekibi tarafından Google'ın iç işlerinde kullanımı için geliştirilmiştir. 2015 yılında Apache License 2.0 sürümü altında piyasaya sürülmüştür.

3.3 Pyqt

PyQt, çapraz platform uygulama geliştirmeye yarayan ve C++ ile yazılmış olan Qt kütüphanesinin Python bağlamasıdır. Python ile grafiksel kullanıcı arayüzlü programlar oluşturmamızı sağlar.

PyQt, İngiliz Riverbank Computing firması tarafından geliştirilmiş ücretsiz bir yazılımdır.

3.4 Matplotlib

Veri görselleştirmesinde kullanılan bir python kütüphanesidir. 2 ve 3 boyutlu çizimler yapmamızı sağlar.

Projede modelin başarı, kayıp grafiklerinin çiziminde kullanılmıştır.

3.5 Seaborn

Seaborn, Matplotlib kütüphanesi tabanlı, istatiksel bir Python veri görselleştirme kütüphanesidir. Seaborn kullanıcılara istatiksel görselleştirmeler yapmaları için high-level (yüksek seviyeli) bir arayüz sunar.

Projede modelin karışıklık matrisinin çiziminde kullanılmıştır.

3.6 NumPy

Python programlama dili için büyük, çok boyutlu dizileri ve matrisleri destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler ekleyen bir kitaplıktır.

Projede veri setinin ön işleme işlemlerinde, karışıklık matrisi oluşturmada ve model tahmin verilerinin işlenmesinde kullanılmıştır.

3.7 OpenCV

Gerçek-zamanlı bilgisayar görüsü uygulamalarında kullanılan açık kaynaklı kütüphane. İlk olarak Intel tarafından geliştirilmiş, daha sonra Willow Garage ve sonra Itseez tarafından Sürdürülmüştür. Bu kütüphane çoklu platform ve BSD lisansı altında açık kaynaklı bir yazılımdır.

Projede Görsellerin işlenmesi için kullanılmıştır.

4. MODEL GELİŞTİRME AŞAMALARI

4.1 VERİ SETİ ÖN İŞLEME

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

- Veri seti tesorflow un veri seti arşivinden alınmıştır.

- MNIST veri setinde train setinin shape ine bakıldığında

```
Training shape:  (60000, 28, 28)
Label shape:  (60000,)
```

Yapısı görünüyor yani buradan eğitim setinde 60.000 tane, 28x28 lik dizi halinde bulunan görüntü olduğu anlaşılıyor. Label dada her bir görüntünün hangi sınıfta olduğunu belirten sınıf numaraları bulunuyor (0-9 arası). Test seti için değişen tek şey 60.000 adet olan görsel sayısı 10.000 oluyor.

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
```

- Görsellerin modele verilirken kanal değerlerinin verilmesi gerekiyor. Bu değer grayscale görseller için 1 iken RGB görseller için 3 oluyor. MNIST veri setindeki görseller grayscale olduğundan modelin eğitimi düzgün yapabilmesi için orijinal x_train ve x_test shape lerine reshape yapılarak 1 verliyor. Bu işlem sonucunda bir görselin shape i (28,28) den (28,28, 1) oluyor. Bu durum sonucunda bir görsel 28 satır 28 sütunluk bir matrisden, 28 tane 28 satır 1 sütunluk matrislere dönüşüyor. “input_shape” değişkeni de modelin ilk katmanına verilerin nasıl yapıda olduğunu belirtmek için verilecektir.

Reshape den önce bir görselin yapısı:

```
Image shape:  (28, 28)
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
```

Reshape den sonra bir görselin yapısı:

```
Image shape:  (28, 28, 1)
[[[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
```

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

x_train /= 255
x_test /= 255
```

- Gayscale pixel değerlerin sayısal ağırlığını azaltmak için 0-255 arasında olan değerleri 0-1 arasına çekerek normalizasyon yapılmıştır. Normalizasyon işlemi 255 bölerek olduğundan pixellerdeki bilgilerinin korunması için bölme işleminden önce tüm değerler “float32” tipine dönüştürülmüştür.

Normalizasyondan önce:

```
  [  0]]

 [[  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  0]
  [  3]
  [ 18]
  [ 18]
  [ 18]
  [126]
  [136]
  [175]
  [ 26]
  [166]
  [255]
  [247]
  [127]
  [  0]
  [  0]
  [  0]
  [  0]]
```

Normalizasyondan sonra:

```
[[0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.01176471]
 [0.07058824]
 [0.07058824]
 [0.07058824]
 [0.49411765]
 [0.53333336]
 [0.6862745 ]
 [0.10196079]
 [0.6509804 ]
 [1.        ]
 [0.96862745]
 [0.49803922]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]]
```

4.2 MODEL OLUŞTURMA

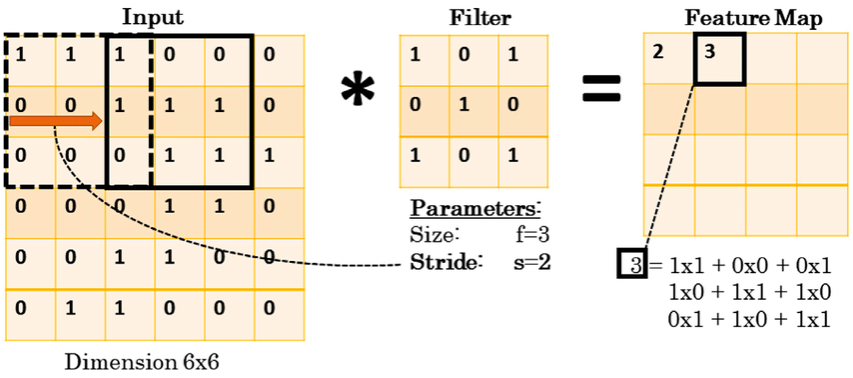
```
model = Sequential()
```

- Modelleri sıralı olarak katman katman oluşturulmasını sağlar. 1 girişi ve 1 çıkışı vardır.

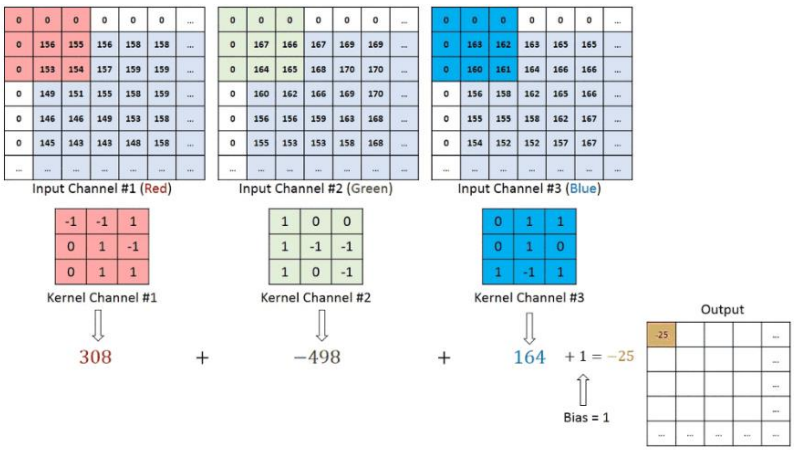
```
model.add(Conv2D(32, (5, 5), input_shape=input_shape, activation='relu'))
```

- 32 : Özellik(Aktivasyon) haritası sayısıdır. Eğitim sırasında verilen verilerin üzerinde belirtilen boyuttaki kernel in gezmesiyle görüntünün farkı özelliklerinin öğrenilmesi için haritalar oluşturulmaya başlanır. İşlem sonucunda her bir filtre 2D özellik haritası üretmiş olur. Kernel boyutları (1,1), (3,3), (5,5) (7,7) gibi olabilir.

Kernel(Filtre) nin verilen tek kanallı veride gezmesi ve özellik haritasının çıkarılması:

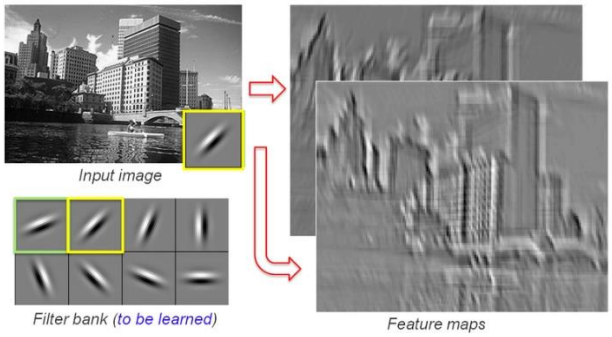


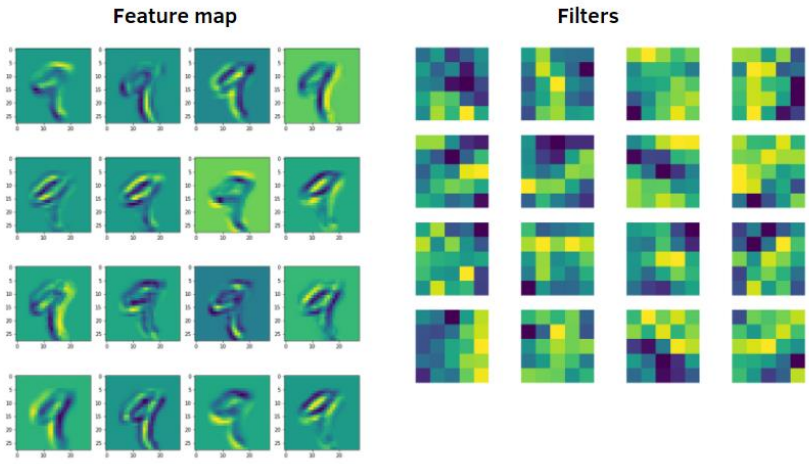
RGB görseller için özellik haritası çıkarımı:



Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

Örnek özellik haritaları:





- (5, 5): Kernel boyutu. Özellik (Aktivasyon) haritaları çıkarma için hesaplamalarda kullanılacak olan kernel in boyutunu belirtir. Tek sayı uzunluğunda kare matris olması önerilir. Sayı değerleri otomatik olarak oluşturulur.

Örnek kernel ler:

3x3

0.91	0.32	0.07
0.73	0.26	0.81
0.53	0.68	0.14

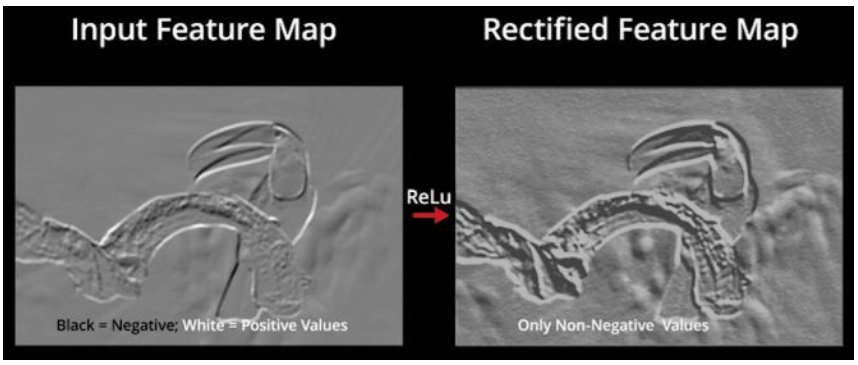
5x5

0.27	0.64	0.44	0.84	0.29
0.28	0.06	0.89	0.99	0.33
0.64	0.67	0.08	0.38	0.03
0.04	0.31	0.16	0.57	0.08
0.87	0.85	0.97	0.71	0.96

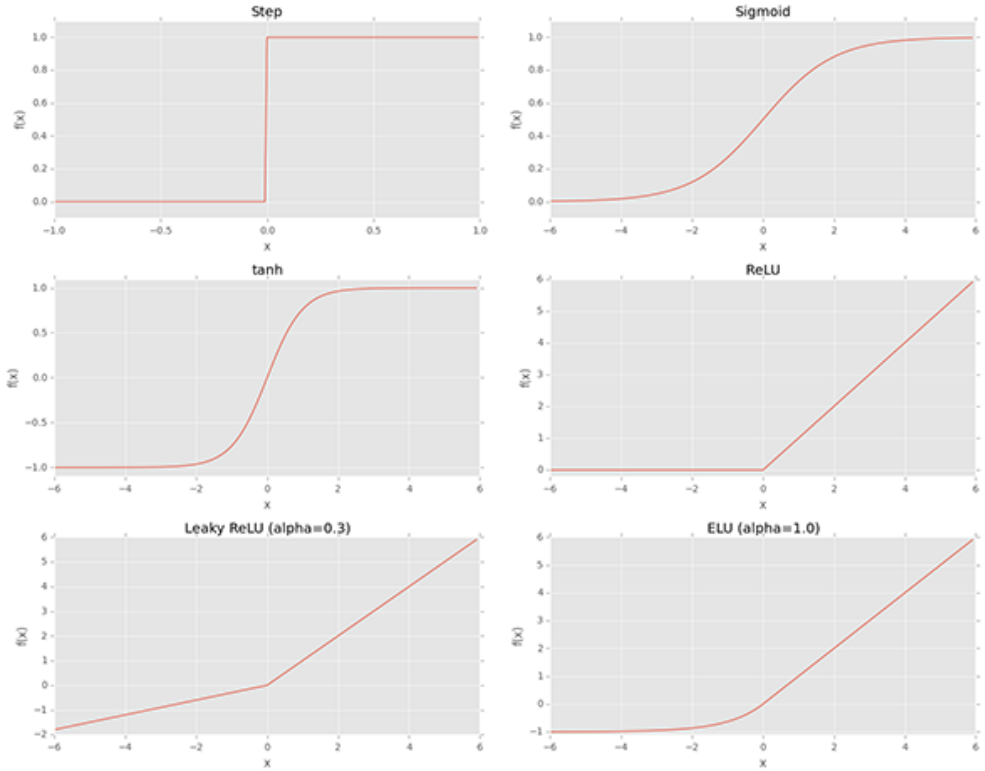
- input_shape: Hesaplamalarda kullanılacak verilerin(görsellerin) yapısını gösteriyor. Projede kullanılan değer (28, 28, 1) yani her görsel tek kanallı 28x28 matrislerden oluşuyor.

- activation: Etkinleştirme parametresi. Yalnızca bir kolaylık parametresidir ve evrişimi gerçekleştirdikten sonra uygulamak istediğimiz etkinleştirme işlevinin adını belirtmeye olanak tanır.

Özellik haritasının relu fonksiyonuna verilmesi:



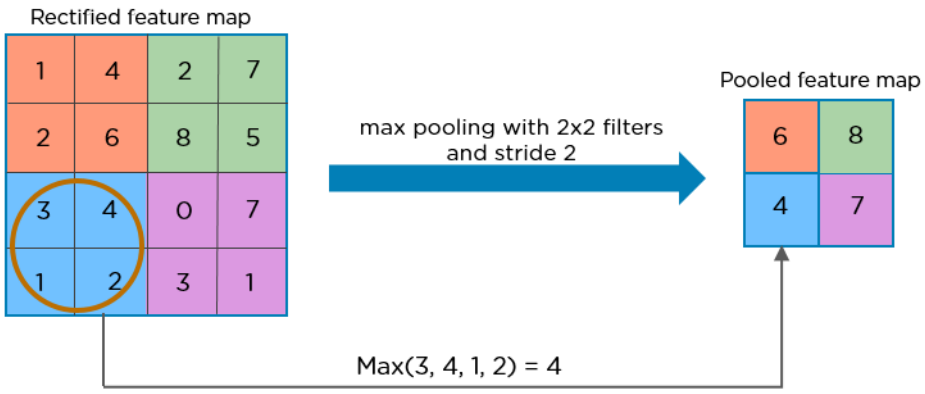
Sık kullanılan aktivasyon fonksiyonlarının grafikleri:



```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

- pool_size: Filtre matrisi boyutunu belirtir. Girişin her kanalı için bir giriş penceresi üzerinden maksimum değeri alarak girişi uzamsal boyunca alt örnekler. Pencere, her bir boyut boyunca adım adım kaydırılır. Önceki evrişim katmanından gelen çıktıdaki piksel sayısını azaltarak görüntülerin boyutsallığını azaltır.

İşleyiş örneği:



- Maksimum havuzlama, bir evrişim katmanının verilen çıktısının çözünürlüğünü azalttığından, ağ, ileri bir zamanda görüntünün daha geniş alanlarına bakacaktır, bu da ağdaki parametre miktarını azaltır ve sonuç olarak hesaplama yükünü azaltır.

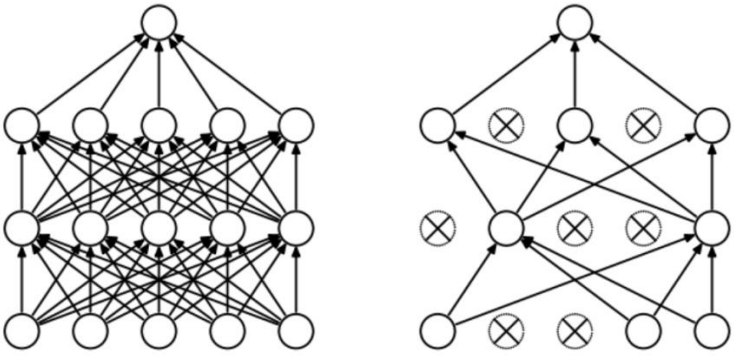
```
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

- Bir Conv2D katanından ve MaxPooling den geçirilen model bir kez daha kernel ı 3x3 olan Conv2D katmanından geçirliyor ve Max pooling uygulanıyor. Veri matris boyutları daha da azalıyor.

```
model.add(Dropout(0.2))
```

- Dropout: Kaldırılacak özellik haritası yüzdesi. Bir modelin aşır öğrenmesini önlemek için kullanılan bir tekniktir. Eğitim aşamasında rasgele özellik haritalarını kaldırır.

Rasgele nöron kaldırma:

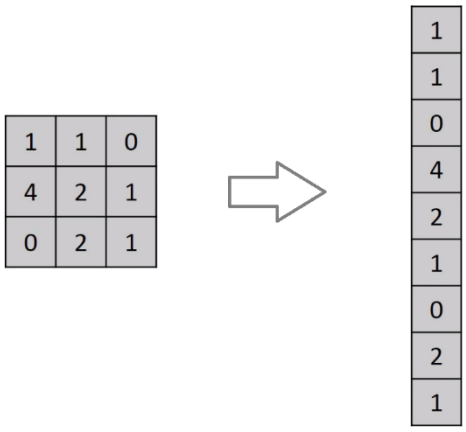


(a) Standard Neural Net (b) After applying dropout.

```
model.add(Flatten())
```

- Katman düzleştirme işlemi. Çok boyutlu girdi tensörlerini tek bir boyuta düzleştirir, böylece girdi katmanını modelleyebilir ve sinir ağı modeli oluşturabilir, ardından bu veriler modelin her bir nöronuna iletilir.

Örnek düzleştirme işlemi:

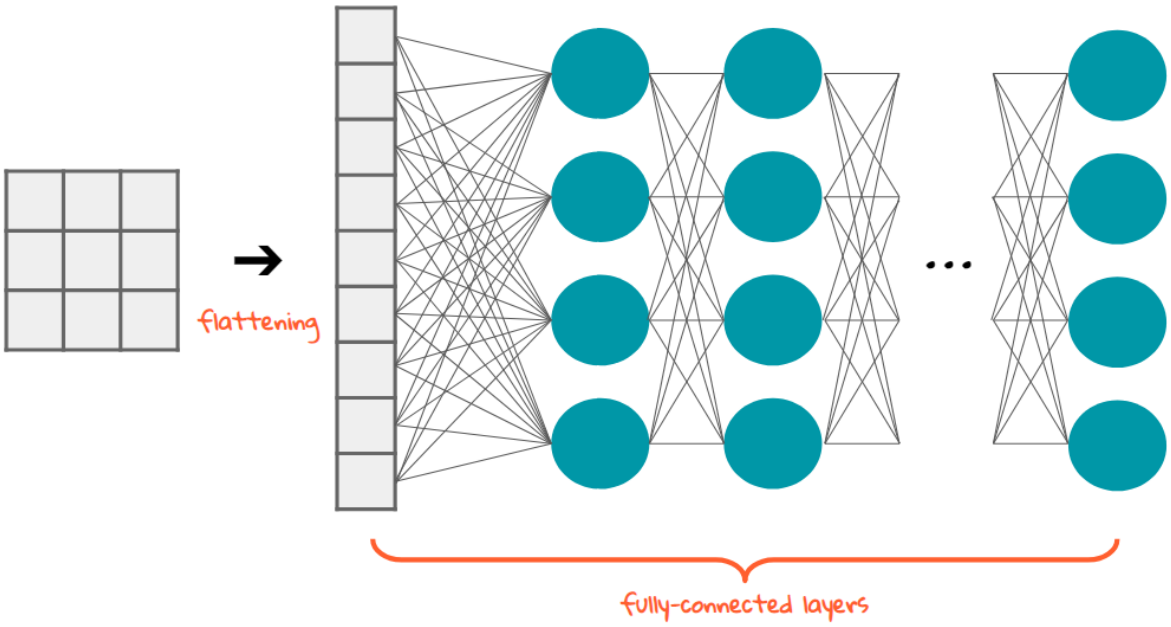



```
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

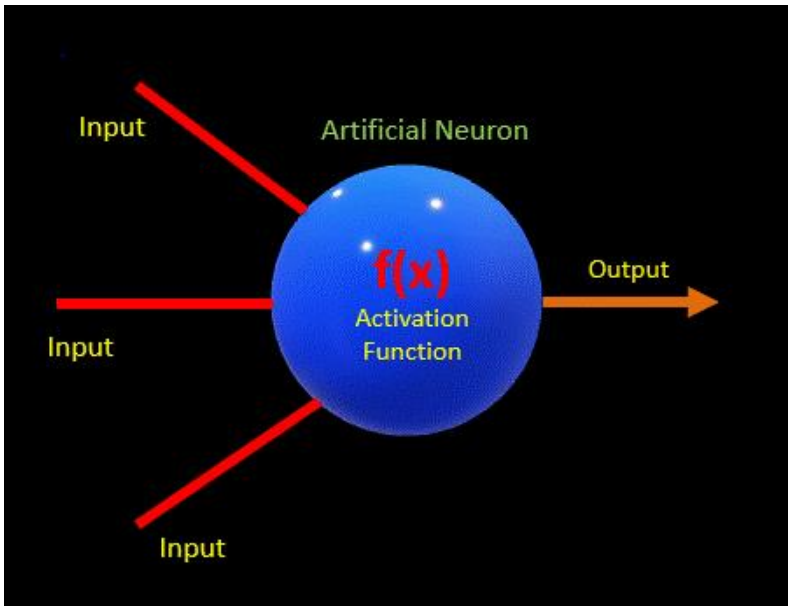
- 128: Nöron sayısı. Düzleştirme işleminden sonra veriler belirtilen sayı kadar nöronlara aktarılıyor. Nöronların aktifleşme durumlarının hesaplanması için “relu” aktifleştirme fonksiyonu kullanılıyor.
- İkinci katmanda ise sınıf sayısı kadar (10) nöron bulunuyor. İlk katmandaki tüm nöronlar bu 10 nörona bağlı oluyor. Bu 10 nöronun aktifleşme durumlarına göre modelin tahmini anlaşılıyor.

- Bu katmanlarda nöronların hepsi birbiri ile bağlı olduğu için tam bağlı katman (fully connected layer) deniyor.

Tam bağlı katman:

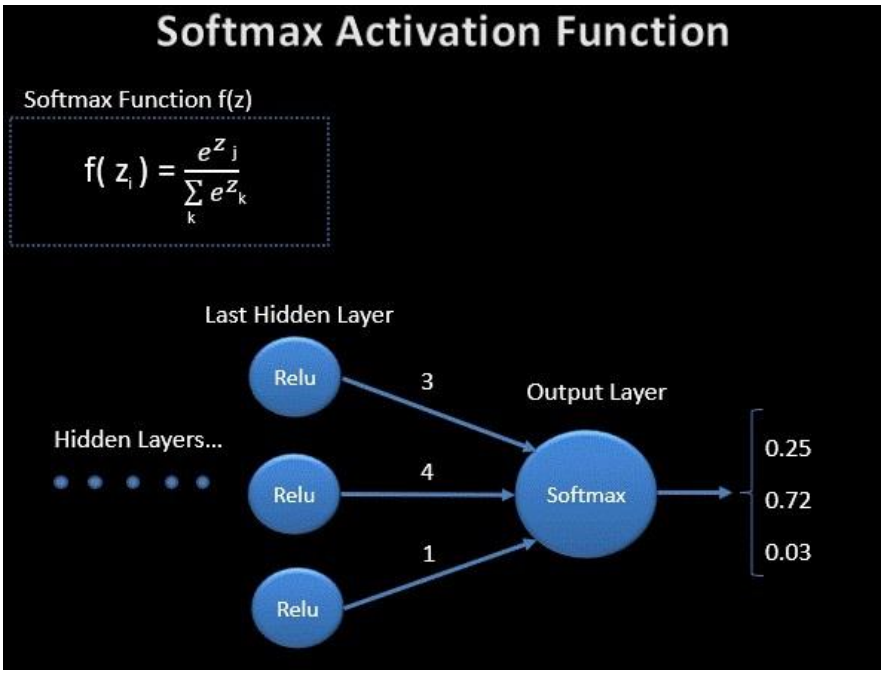


Aktifleştirme fonksiyonu konumu:



- softmax: aktivasyon fonksiyonu. Yapay bir nöronun giriş sinyallerini bir olasılık dağılımına dönüştürür. Genellikle, çıktı olarak sınıflar için olasılık dağılımı üretmemiz gereken çok sınıflı sınıflandırıcılar için sinir ağının son katmanında kullanılır.

Softmax aktivasyon fonksiyonu:




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

- compiler: Modeli eğitim için yapılandırır.
- loss: Bir makine öğrenimi modelinin performansını iyileştirmeyi ve belirli sorunları daha verimli bir şekilde çözmeyi sağlar.
- optimizer: Kayıpları azaltmak için makine öğrenme modelinin ağırlık ve öğrenme oranı gibi özelliklerini değiştirmek için kullanılan Sınıflar veya yöntemlerdir. Sonuçların daha hızlı alınmasına yardımcı olur.
- Adam: Birinci dereceden ve ikinci dereceden momentlerin uyarlanabilir tahminine dayanan stokastik bir gradyan iniş yöntemidir.
- metrics: Eğitim ve test sırasında model tarafından değerlendirilecek metriklerin listesi.
- accuracy: Tahminlerin etiketlere ne sıklıkla eşit olduğunu hesaplar. Başarıyı denetler

```
history=model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10)
```

- history: Model eğitilirken oluşan eğitim verileri daha sonra kullanılmak üzere aktarılıyor.
- fit: Eğitimde kullanılacak veriler ve eğitim parametreleri veriliyor.
- validation_data: Modelin düzgün eğitilip eğitilmediğini test etmek için test seti veriliyor.
- epochs: Modelin kaç kez tüm eğitim seti üzerinden geçerek eğitileceğini belirtiyor.

4.3 MODEL BAŞARI VERİLERİNİN ALINMASI

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Başarı Grafiği')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig("acc.png")


plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Kayıp Grafiği')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig("loss.png")
```

- Model fit işleminden sonra eğitim ile ilgili veriler “history” değişkenine aktarılıyor.
- Değişkenin içindeki ilgili veriler kullanılarak başarı, kayıp grafikleri oluşturuluyor.

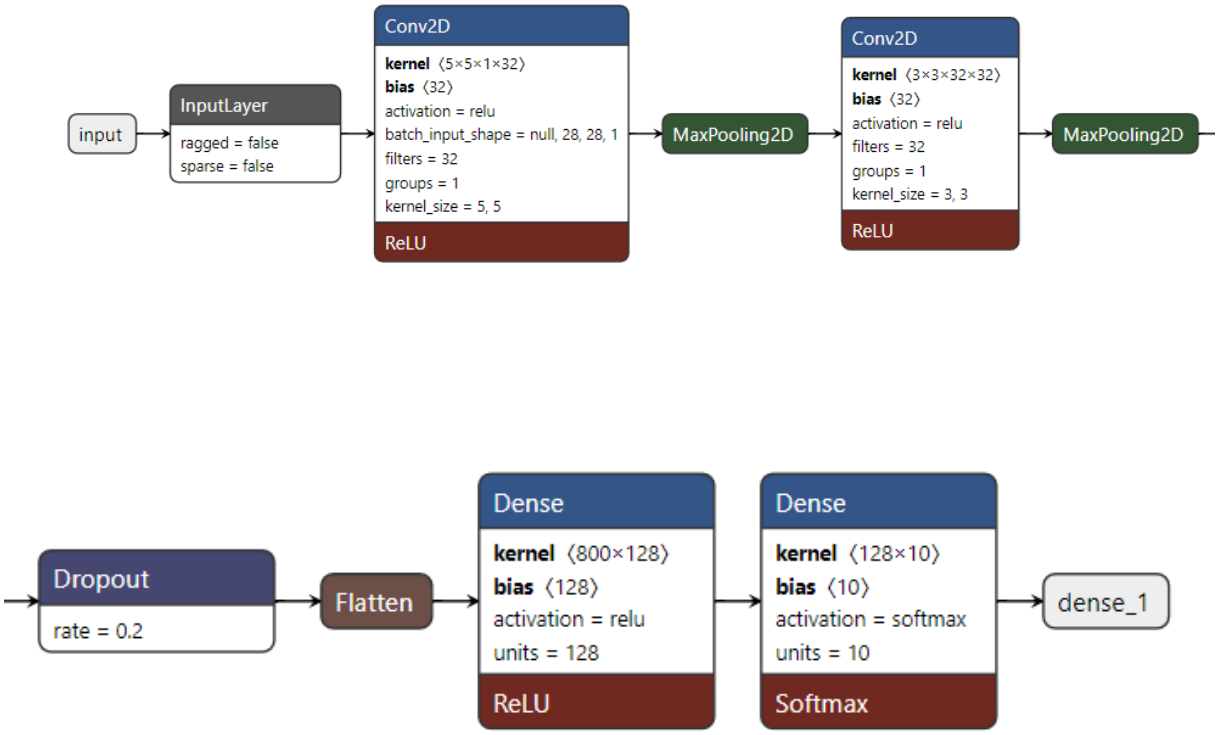
```
test_tahminleri=model.predict(x_test)

test_tahminleri = [np.argmax(i) for i in test_tahminleri]
cm = tf.math.confusion_matrix(labels=y_test, predictions=test_tahminleri)

plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Tahmin edilen')
plt.ylabel('Gerçek')
plt.savefig("cm.png")
```

- Karışıklık matrisinin çizilmesi için test setindeki tüm veriler ile model tek tek tahminde bulunuyor. Bu tahminler işlenip, “confusion_matrix” metoduna gerçek de olması gereken değerler ile birlikte veriliyor ve matris hesaplanıyor.
- İlgili kütüphane ile bilgiler görsel grafiğe dökülüyor.

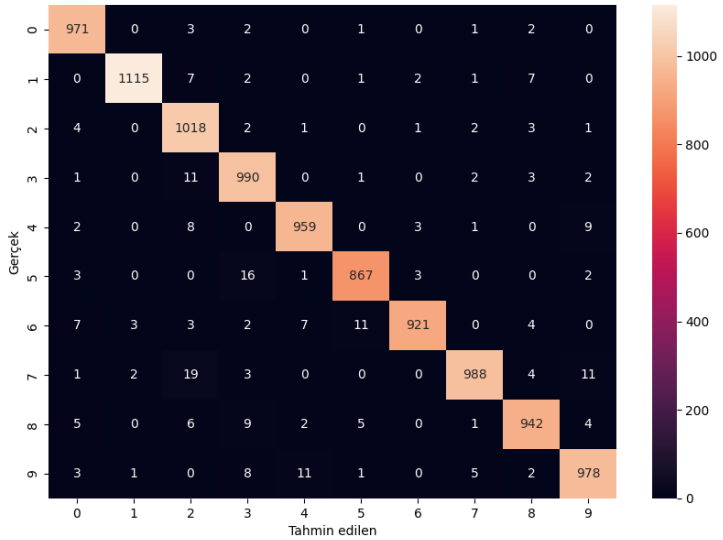
4.4 MODEL İN GÖRSELLEŞTİRİLMİŞ GÖRÜNÜMÜ



4.5 NİHAİ MODEL SEÇİLENE KADARKİ TEST EDİLEN MODELLER

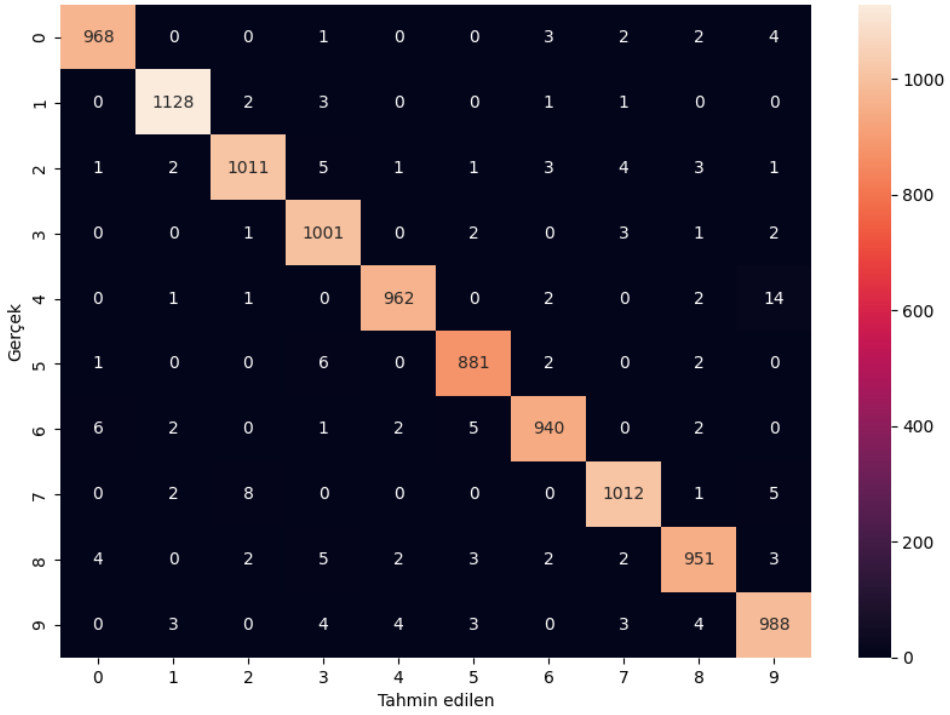
NOT: Tüm modeller için veri seti normalize edilmiştir.

4.5.1 1.Model



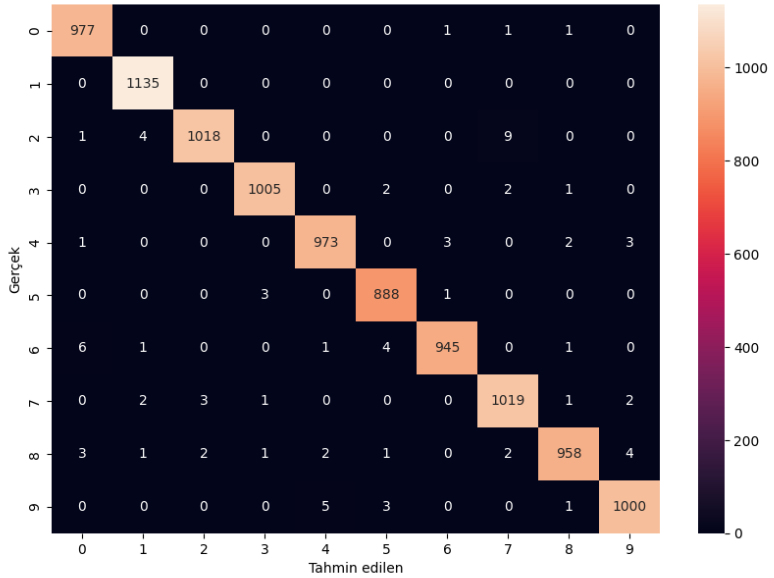
```
model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28,28)),
                                    tf.keras.layers.Dense(128, activation='relu'),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

4.5.2 2.Model



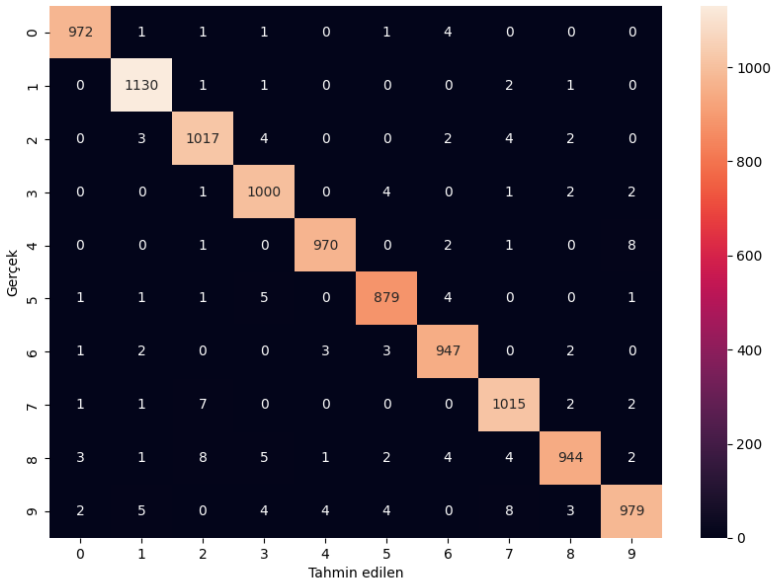

```
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10,activation=tf.nn.softmax))
```

4.5.3 3.Model



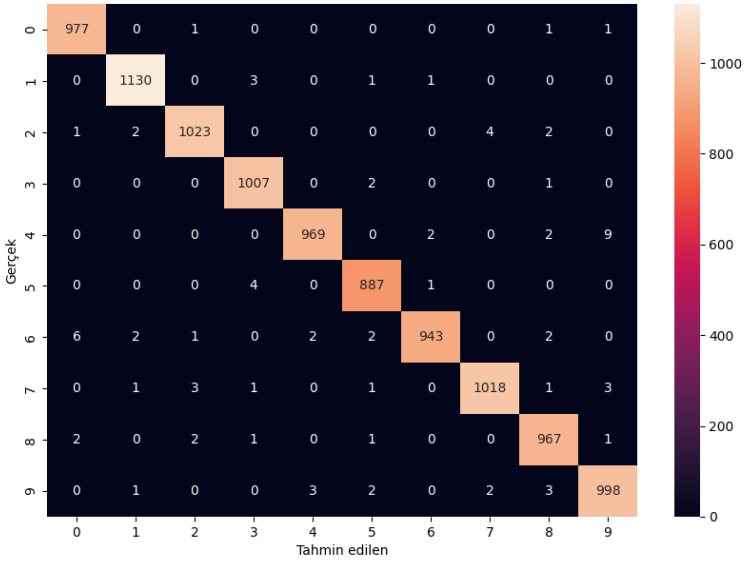

```
model =Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

4.5.4 4.Model



```
model = Sequential()
model.add(Conv2D(25, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(1,1)))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

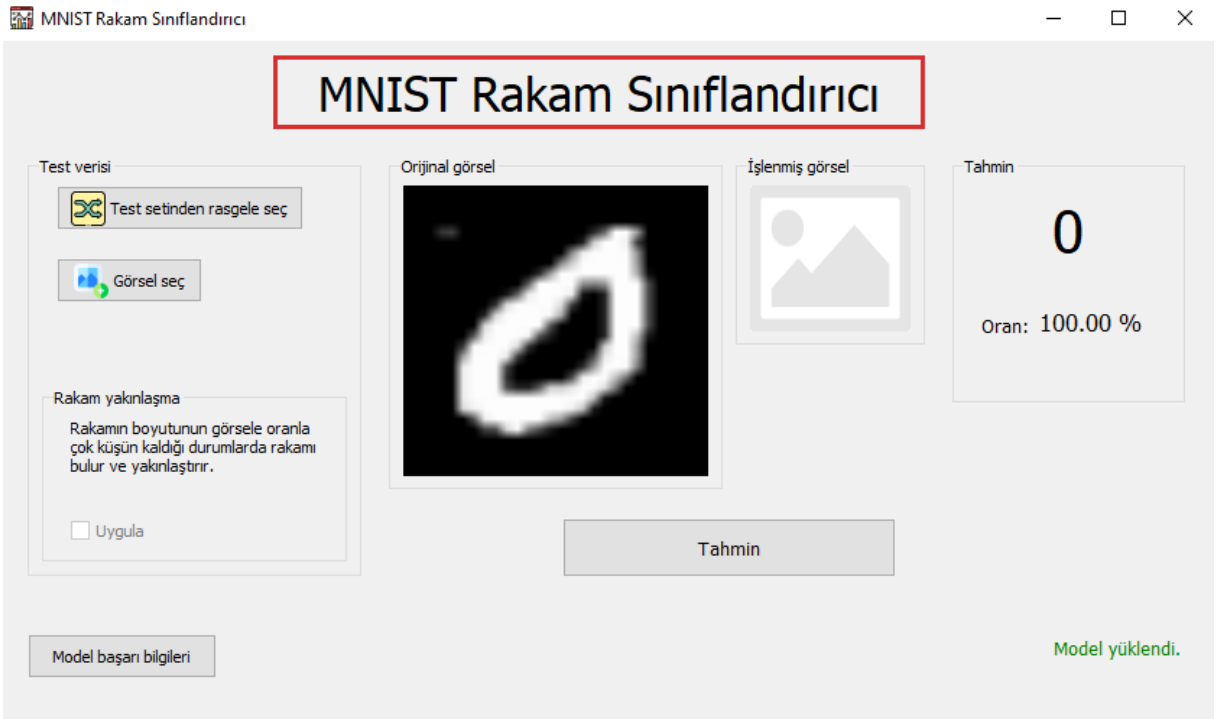
4.5.5 5.Model




```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=input_shape))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(10,activation="softmax"))
```

5 UYGULAMA KULLANIMI

- "Test setinden rasgele seç" butonuna basıldığında test setinden rasgele bir görsel seçiliyor ve "Orijinal görsel" alanında seçilen görsel gösteriliyor.
- "Tahmin" butonuna basıldığında da test setinden seçilen görsel ile model bir tahmin yapıyor.
- Tahmin edilen rakam ve tahmin yüzdesi "Tahmin" groupbox unda gösteriliyor.



- “Görsel seç” butonuna basıldığında dışarıdan bir görsel seçilebiliyor ve otomatik olarak uygun işleme yöntemi uygulanarak işlenmiş hali “İşlenmiş görsel” alanında gösteriliyor.

- Herhangi bir görsel seçilmediği durumda kullanıcıya uyarı veriyor ve tahmin yapmıyor.

- Test işleminde boyut olarak uygun olmayan görseller için “Rakam yakınlaşma” özelliği ile görseldeki rakam otomatik olarak algılanıyor ve görselin yeni hali “İşlenmiş görsel” alanında gösteriliyor.

- “Model başarı bilgileri” butonuna basıldığında modelin bazı başarı bilgileri gösteriliyor.