Evrişimsel Sinir Ağları ile Görüntü Tanıma

Burak Tahtacı, Ömer Faruk Ekuklu

Ocak 2019

1 Yöntem

Evrişim yani kovolüsyon işlemi, bir filtre matrisinin bir resim üzerinde gezdirilerek çarpılması işlemidir. Evrişimsel sinir ağları ise bir girdi görüntüsüne genel özellik çıkarma filtreleri uygulanmaktadır. Bu filtreler vasıtasıyla özellik çıkarımı (feature extraction) işlemi yapılmaktadır. Filtrelerden çıkan özelliklerin sayısı çok fazla olduğu için bu özellikleri indirgemek ve normalize etmek için iki katman daha kullanılmaktadır. Bunlar Max Poolling ve Batch Normalizer katmanlarıdır. Max Pooling bir havuzlama katmanıdır ve özellik indirgeme işlevini yerine getirir. Batch Normalizer katmanı ise çıkan özelliklerin varyanslarını normalize ederek çıkan özelliklerin dağılımını azaltarak daha doğru yorumlanmasını sağlar.

Ardından elde edilen özellikler tam-bağlı(fully connected) katmanlardan geçirilerek sınıflandırma işlemi başlamış olur. Yapılan çalışma kapsındaki veri setinde 10 farklı sınıf bulunmaktadır. Fully Conntected katmanlardan gelen değerler çıktı katmanına aktarılır. Bu katmanda 10 sınıflı bir softmax katmanı bulunmaktadır.

```
model = Sequential()
model.add(Conv2D(32, kernel_size = (3, 3), activation='relu', input_shape=(200, 200, 1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation = 'softmax'))
```

Figure 1: Geliştirilen Modelin Yapısı

Yapay sinir ağı oluşturulurken konvolüsyon katmanları için Keras'ın Conv2D katmanı kullanılmıştır. Aktivasyon fonksiyonları ise relu olarak belirlenmiştir. 5 tane konvolüsyon katmanı ve ardından 2 tane fully conntected katmanla birlikte 1 adet çıktı katmanı kullanılarak model oluşturulmuştur. 0.2 Dropout kullanarak modelin overfit olması engellenmiştir.

2 Uygulama

Çalışma için verilen veri setinde 2169 adet görüntü bulunmaktadır. Bu görüntüler 10 farklı sınıfa aittir. Veri seti train ve test olarak ayrılmadığı için ilk olarak 80'e 20' oranında train ve test veri seti oluşturulmuştur. Ardından verisetindeki her resim 200x200 olacak şekilde yeniden boyutlandırılıp numpy array olarak okunmuştur. Sınıf etiketlerini one-hot-encoded şekilde tutulmuştur.

2.1 Başarı Ölçümü

İnternetten rastgele bulunan örnekler için sistem test edildiğinde bulunan sonuçlar aşağıdaki gibi olmuştur. Her sınıftan 5'er adet görüntü seçilip sistem test edilmiştir. 50 adet görüntüden 21 tanesini sistem doğru olarak sınıflandırmıştır. Sınıflandırma sonuçları aşağıdaki gibidir.

```
File ->  ucak4.jpeg  Prediction -> airplane
File ->  laptop4.jpeg  Prediction -> laptop
File ->  ucak5.jpeg  Prediction -> airplane
File ->  laptop5.jpeg  Prediction -> car_side
File ->  pizza.jpeg  Prediction -> butterfly
File ->  cup.jpeg  Prediction -> dolphin
File ->  headphone2.jpeg  Prediction -> headphone
File ->  pizza3.jpeg  Prediction -> pizza
File ->  motosiklet5.jpeg  Prediction -> Motorbikes
File ->  dolphin.jpeg  Prediction -> airplane
File ->  ucak2.jpeg  Prediction -> airplane
File ->  laptop2.jpeg  Prediction -> cell_phone
File ->  motosiklet2.jpeg  Prediction -> headphone
File ->  headphone.jpeg  Prediction -> laptop
File ->  cell.jpeg  Prediction -> airplane
File ->  pizza4.jpeg  Prediction -> Motorbikes
File ->  car.jpeg  Prediction -> airplane
File ->  headphone5.jpeg  Prediction -> cell_phone
File ->  kelebek.jpeg  Prediction -> butterfly
File ->  headphone4.jpeg  Prediction -> cup
File ->  pizza5.jpeg  Prediction -> pizza
File ->  ucak3.jpeg  Prediction -> airplane
File ->  laptop3.jpeg  Prediction -> airplane
File ->  kelebek4.jpeg  Prediction -> Motorbikes
File ->  dolphin5.jpeg  Prediction -> airplane
File ->  cup5.jpeg  Prediction -> cup
File ->  cell2.jpeg  Prediction -> butterfly
File ->  laptop.jpeg  Prediction -> airplane
File ->  car5.jpeg  Prediction -> airplane
File ->  ucak.jpeg  Prediction -> airplane
File ->  motosiklet1.jpeg  Prediction -> airplane
File ->  car4.jpeg  Prediction -> Motorbikes
File ->  cell3.jpeg  Prediction -> laptop
File ->  cup4.jpeg  Prediction -> airplane
File ->  dolphin4.jpeg  Prediction -> dolphin
File ->  kelebek5.jpeg  Prediction -> cup
File ->  cell4.jpeg  Prediction -> airplane
File ->  car3.jpeg  Prediction -> cup
File ->  kelebek2.jpeg  Prediction -> butterfly
File ->  dolphin3.jpeg  Prediction -> butterfly
File ->  cup3.jpeg  Prediction -> cup
File ->  cup2.jpeg  Prediction -> cup
File ->  dolphin2.jpeg  Prediction -> dolphin
```

Figure 2: Validasyon Sonuçları

2.2 Data Augmentation

Görüntü işleme uygulamalarında verisetindeki görüntüler farklı açılarda çekilmiş olabilir. Tanıma yaparken farklı açılarda gelen görüntüleri de tanımak amacıyla veri çoğaltma (data augmentation) işlemi uygulanmaktadır. Bu çalışma kapsamında fotoğraflar 90'ar derece sağa ve sola yatırılarak veri seti çoğaltılmıştır. Bu işlem sistemin genel performansını artırmıştır.

3 Sonuç

Modelin train ve test işlemleri Intel Core i5 CPU üzerinde 8GB bellekli bir makinede gerçekleştirilmiştir. Bilgisayar kaynakları yetersiz olduğundan batch size 50 iken 2 epoch eğitim yapılmıştır. Train işleminin sonuçları aşağıdaki gibidir.

```
model.fit(trainImages,trainLabels, batch_size = 50, epochs = 2, verbose = 1)
```

```
Epoch 1/2
1736/1736 [==============================] - 141s 81ms/step - loss: 0.1375 - acc: 0.9565
Epoch 2/2
1736/1736 [==============================] - 136s 78ms/step - loss: 0.0656 - acc: 0.9768
```

Figure 3: Eğitim Sonuçları

Sistemin test veri seti ile sınanmasının ardından elde edilen sonuçlar aşağıdaki gibidir.

```
test_loss, test_acc = model.evaluate(testImages, testLabels)

print('Test accuracy:', test_acc)
```

```
433/433 [==============================] - 9s 22ms/step
Test accuracy: 0.9762124720676942
```

Figure 4: Test Sonuçları