Rice Image Classification

NEURAL NETWORKS Dersi Proje Rapor

Github: https://github.com/cagridogan08/RiceImageClassification

152120151071 Çağrı Doğan
Bilgisayar Mühendisliği

***Özet*— Dünyada en çok üretilen tahıl ürünleri arasında yer alan pirinç, birçok genetik çeşide sahiptir. Bu çeşitler bazı özelliklerinden dolayı birbirinden ayrılmaktadır. Bunlar genellikle doku, şekil ve renk gibi özelliklerdir. Pirinç çeşitlerini birbirinden ayıran bu özellikler ile tohumların sınıflandırılması ve kalitesinin değerlendirilmesi mümkündür. Bu çalışmada Türkiye'de sıklıkla yetiştirilen beş farklı pirinç çeşidi olan Arborio, Basmati, İpsala, Yasemin ve Karacadağ kullanılmıştır. Veri setinde çeşitlerin her birinden 15.000 olmak üzere toplam 75.000 tane görüntü bulunmaktadır.**

Bu projede Evrişimsel Sinir Ağı (CNN) algoritması kullanılarak model oluşturulmuş ve sınıflandırma işlemleri gerçekleştirilmiştir. Yapılan sınıflandırma işlemi sonucu model değerlendirme metrikleri görsel olarak verilmiştir.

I. GİRİŞ

Bir evrişimsel sinir ağı (ConvNet / Convolutional neural networks -CNN), bir girdi görüntüsünü alıp, görüntüdeki çeşitli görünüşleri/nesneleri birbirinden ayırabilen derin öğrenme algoritmasıdır.[1] Evrişimli sinir ağları, temel olarak görüntüleri sınıflandırmak (örneğin gördüklerini isimlendirmek), benzerlikle kümelemek (fotoğraf arama) ve sahnelerde nesne tanıma yapmak için kullanılan derin yapay sinir ağlarıdır.

Bu projede Resim Veri Kümesi Yükleme(ImageDataGenerator) ile train ve validation veri grupların(batch) oluşturmat, sininr ağı modeli inşaası, derlenmesi ve eğitimi gerçekleştirilmiştir.

II. VERI KUMESI

A. *Verileri Okuma*

Derin öğrenme sinir ağı modellerini daha fazla veri üzerinde eğitmek, daha yetenekli modellerle sonuçlanabilir ve büyütme teknikleri, uygun modellerin öğrendiklerini yeni görüntülere genelleştirme yeteneğini geliştirebilen görüntülerin varyasyonlarını oluşturabilir.

Keras derin öğrenme sinir ağı kitaplığı, "ImageDataGenerator" sınıfı aracılığıyla görüntü verisi büyütmeyi kullanarak modelleri uydurma yeteneği sağlar.[2]

ImageDataGenerator görüntüler için bir üreticidir (generator) ve gerçek zamanlı veri çeşitlendirme (real-time data augmentation) yaparak görüntü verilerini yığınlar(batch) olarak oluşturur.

```
image_gen = ImageDataGenerator(
    rescale = 1.0/255.0,#normalization
    validation_split=0.2 # train-validation split
)
train_batches = image_gen.flow_from_directory(
    directory = rice_path,
    target_size=(128,128),#image size
    batch_size = 32,
    subset='training',
    class_mode='categorical'
)
validation_batches = image_gen.flow_from_directory(
    directory = rice_path,
    target_size = (128,128),
    batch_size = 32,
    subset = 'validation',
    class_mode='categorical'
)
```

Şekil 1- Verinin Ön işlenmesi ve Okunması

Şekil 1'de bulunan kod parçasında ImageDataGenerator fonksiyonuna girdi olarak verilen "rescale" değeri görüntü verisinin normalleştirilmesi için, validation_split değeri ise verinin %80 train- %20 validation işlemleri için kullanılmak üzere bölünmesi amaçlanmıştır.

flow_from_directory bir ImageDataGenerator metodudur. Bu methodun çıktısı (x, y) demetlerini veren bir DirectoryIterator'dır. Burada, x (yığın büyüklüğü, *hedef büyüklüğü, kanallar) şekline sahip bir görüntüler yığını olan NumPy dizisidir ve y bu görüntülere karşılık gelen etiketlerin NumPy dizisidir.[3]

directory değeri verinin bulunduğu klasör

target_size: görüntünün yükseklik ve genişlik değerleri

batch_size: yığın büyüklüğüdür ve varsayılan olarak 32'dir.

Subset: verilerin alt kümesinden eğitim ve doğrulama kümeleri oluşturmak için kullanılır (alacağı değerler training veya validation).

```
Found 60000 images belonging to 5 classes.
Found 15000 images belonging to 5 classes.
```

Görüldüğü üzere 75000 pirinç görüntüsünün 60000 tanesi (%80) eğitim kümesi için, 15000 tanesi (%20) ise doğrulama kümesi için ayrılmıştır.

B. *Veri Kümesinin İncelenmesi*

Şimdi eğitim setinden bir grup görüntü ve etiket oluşturmak için next(train_batch) öğesini çağırıyoruz.

```
train_batches.class_indices
```

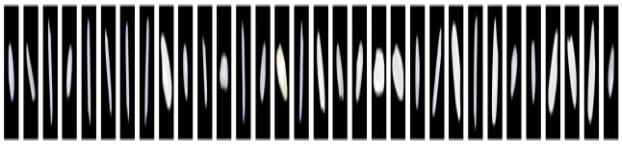
```
{'Arborio': 0, 'Basmati': 1, 'Ipsala': 2, 'Jasmine': 3, 'Karacadag': 4}
```

```
imgs,labels = next(train_batches)
print(imgs.shape)
print(labels.shape)
```

```
(32, 128, 128, 3)
(32, 5)
```

Şekil 3

Şekil 3'te görüldüğü üzere, TensorFlow, elimizdeki dizinden 32 tane resim seçecektir çünkü batch_size = 32 olarak ayarlanmıştır ve her bir görüntünün boyutu (128, 128, 3) 'dir. class_mode='categorical' olarak ayarlandığı için etiketler kategorik vektörler olarak yani bir-elemanı-bir olan vektörler olarak kodlanmıştır. Bu nedenle her bir resmin etiketinin boyutu 5'tir, çünkü elimizde 5 sınıf vardır.



Şekil 4-Eğitim setinden işlenen ilk rastgele yığın

```
array([[0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1.]], dtype=float32)
```

Şekil 5-Görüntüleri bulunan yığının label değerleri

III. MODEL

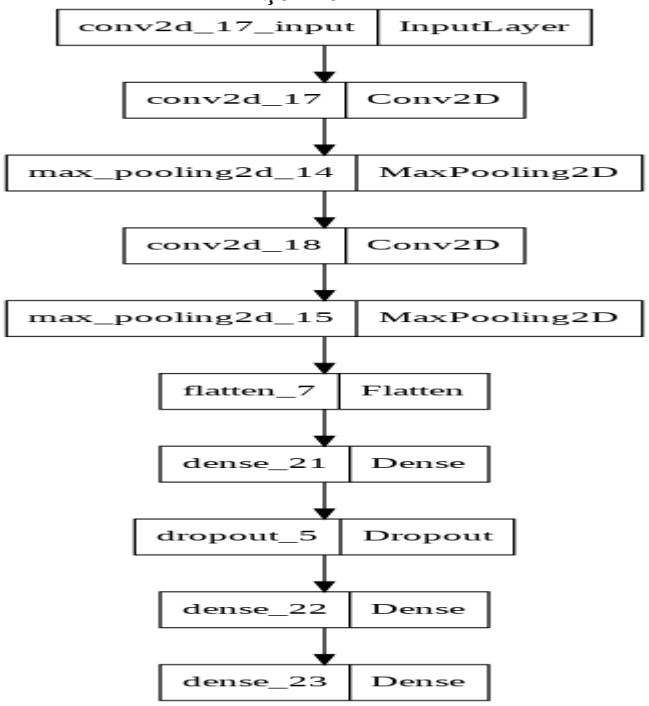
A. *Modelin Oluşturulması*

Keras, python'da sinir ağları oluşturmak için üst düzey bir API'dir.
Bu projede ardışık(sequential) bir model oluşturulacaktır. Modelin mimarisi şu şekildedir:

Input Layer (input_shape= (128,128,3)-> Conv2D (16, (3,3),'relu')-> MaxPool2D (4,4)-> Conv2D(32,(3,3),'relu')-> MaxPool2D(2,2)-> Flatten-> Dense(64,'relu')-> Dropout(0.5)-> Dense(32,'relu')-> Dense(5,'softmax') (Output Layer)

```
model = Sequential()
model.add(Conv2D(16,kernel_size=(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPool2D(4,4))
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32,activation='relu'))
model.add(Dense(5,activation='softmax'))
```

Şekil 6



Şekil 7- Model Mimarisi

B. *Modelin Derlenmesi ve Eğitilmesi*

Modelin derlenmesi için optimizasyon algoritması olarak adaptif momentum sağladığı için "Adam" optimizer, kayıp fonksiyonu olarak ise çoklu sınıflandırma işlemi gerçekleştirileceği için "Categorical Cross-Entropy Loss" seçilmiştir. Metric değeri olarak ise 'accuracy' değeri verilmiştir.

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

Model bir kez derlendikten sonra eğitilmeye başlanabilir. Modeli eğitmede kullanılacak verinin yanısıra iki önemli parametre daha fit () fonksiyonuna verilmedilir. Bunlar bir seferde kullanılacak veri noktası sayısı (batch size) ve verinin üzerinde kaç iterasyon yapılacağı (epoch) verileridir.

```
model.fit(train_batches,batch_size=32,epochs=3,validation_data=validation_batches)
```

Şekil 8- Model Fit

Training Data: train_batches -> eğitim verisi ve labellarının bulunduğu iterator
Batch_Size:32 -> Verilerin okunması aşamasında verilen değerdir
Epochs: 3-> Modelin 3 kez iterasyon yapacağını belirtir.
Validation_data: validation_batches -> doğrulama verisi ve labellarının bulunduğu iterator

IV. Model Performans Metrikleri

Bu kısımda modelin doğruluk oranları ve hata payı üzerinden değerlendirmeler yapılacaktır.

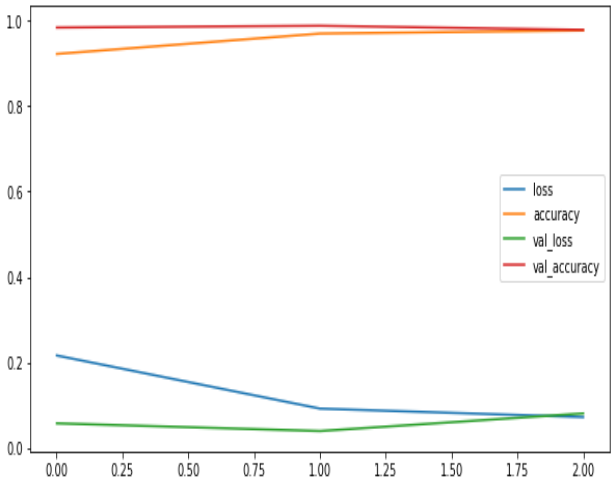
Accuracy: Accuracy değeri modelde doğru tahmin ettiğimiz alanların toplam veri kümesine oranı ile hesaplanmaktır.[4]

Training Accuracy: Modelin eğitim verilerinin üzerindeki doğruluk oranıdır.

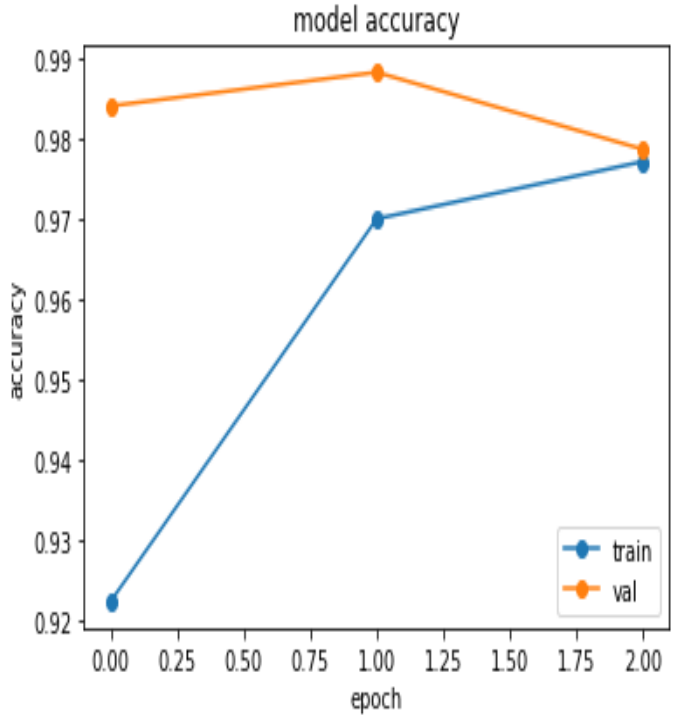
Val_Accuracy: Test doğruluğu, bir modelin görmediği örnekler üzerindeki doğruluğudur.

Training_Loss: Bir derin öğrenme modelinin eğitim verilerine nasıl uyduğunu değerlendirmek için kullanılan bir ölçümdür.

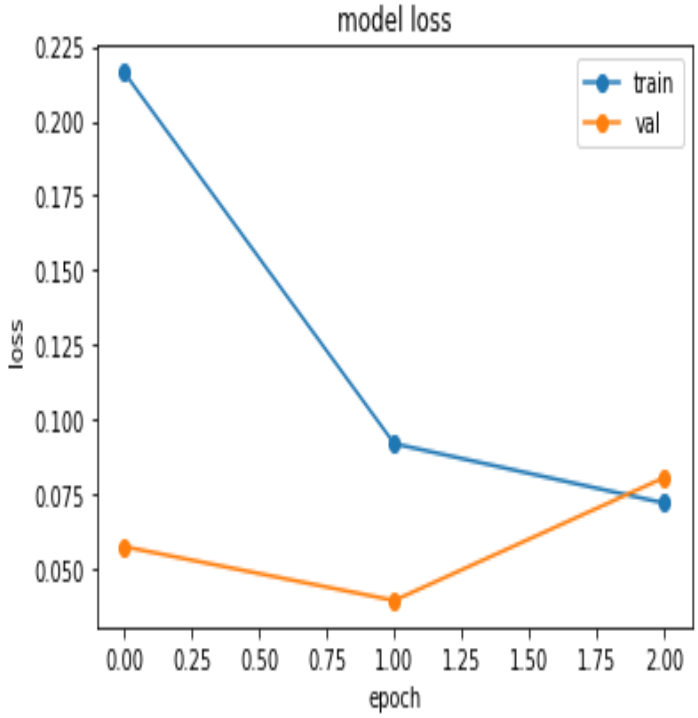
Validation Loss: Bir derin öğrenme modelinin doğrulama setindeki performansını değerlendirmek için kullanılan bir ölçümdür.



Şekil 9- Modeli Accuracy ve Loss Grafiği



Şekil 10-Model Eğitim ve Val. Doğruluğu



V. Kaynakça

[1] https://mustafaserdarkonca.medium.com/evri%C5%9Fimli-sinir-a%C4%9Flar%C4%B1-convolutional-neural-networks-cnn-74b3d4a567f9
[2],[3]- https://mmuratarat.github.io/turkish/2021-03-15/reading-images-into-TF

[4]-
https://medium.com/@gulcanogundur/do%C4%9Fruluk-accuracy-kesinlik-precision-duyarl%C4%B1l%C4%B1k-recall-ya-da-f1-score-300c925feb38
---
https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks
https://www.jeremyjordan.me/convolutional-neural-networks/

https://medium.com/@rabiaokumus96/convolutional-neural-networks-evri%C5%9Fimsel-sinir-a%C4%9Flar%C4%B1-cceb887a2979

https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4